Reference Manual

Allen-Bradley

# Logix 5000 Controllers Import/Export

1756 ControlLogix, 1756 GuardLogix, 1769 CompactLogix, 1769 Compact GuardLogix,
1789 SoftLogix, 5069 CompactLogix, 5069 Compact GuardLogix, Studio 5000 Logix Emulate

# Important user information

Read this document and the documents listed in the additional resources section about installation, configuration, and operation of this equipment before you install, configure, operate, or maintain this product. Users are required to familiarize themselves with installation and wiring instructions in addition to requirements of all applicable codes, laws, and standards.

Activities including installation, adjustments, putting into service, use, assembly, disassembly, and maintenance are required to be carried out by suitably trained personnel in accordance with applicable code of practice. If this equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

In no event will Rockwell Automation, Inc. be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, Rockwell Automation, Inc. cannot assume responsibility or liability for actual use based on the examples and diagrams.

No patent liability is assumed by Rockwell Automation, Inc. with respect to use of information, circuits, equipment, or software described in this manual.

Reproduction of the contents of this manual, in whole or in part, without written permission of Rockwell Automation, Inc., is prohibited.

Throughout this manual, when necessary, we use notes to make you aware of safety considerations.

| | |
|---|---|
| | **WARNING:** Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss. |
| | **ATTENTION:** Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence |
| **Important:** | Identifies information that is critical for successful application and understanding of the product. |

Labels may also be on or inside the equipment to provide specific precautions.

| | |
|---|---|
| | **SHOCK HAZARD:** Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present. |
| | **BURN HAZARD:** Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures. |
| | **ARC FLASH HAZARD:** Labels may be on or inside the equipment, for example, a motor control center, to alert people to potential Arc Flash. Arc Flash will cause severe injury or death. Wear proper Personal Protective Equipment (PPE). Follow ALL Regulatory requirements for safe work practices and for Personal Protective Equipment (PPE). |

Allen-Bradley, Rockwell Software, Rockwell Automation, and TechConnect are trademarks of Rockwell Automation, Inc.

Trademarks not belonging to Rockwell Automation are property of their respective companies.

This manual includes new and updated information. Use these reference tables to locate changed information.

Grammatical and editorial style changes are not included in this summary.

### Global changes

None in this release.

### New or enhanced features

This table contains a list of topics changed in this version, the reason for the change, and a link to the topic that contains the changed information.

| Topic Name | Reason |
|---|---|
| Supported controllers on page 18 | Updated the list of supported controller models. |
| Axis tag attributes on page 119 | Updated the list of Axis tag attributes. |
| Functions on page 187 | Added the function element. |

# Table of contents


**Preface**

Studio 5000 environment ... 17
Supported controllers ... 18
Additional resources ... 20
Legal Notices ... 21

## Chapter 1

**Import and export files**

Introduction ... 25
Export a Project to an.L5K Text File ... 25
Import an.L5K text file ... 26
Export a Project to an.L5X XML File ... 28
Import an.L5X XML File ... 29
Export to a.CSV or.TXT file ... 30
Import a.CSV or.TXT file ... 32
Export source-protected logic ... 34
    Export in a Cleartext Format ... 35
Maintaining controller access ... 37
.L5X file structure ... 38
.L5X file conventions ... 41
    Internal file comments ... 41
    Component Descriptions ... 42
    Boolean attribute values ... 42
    Data display style ... 42
    Data formats ... 43
.L5K file structure ... 47
.L5K file conventions ... 48
    Internal file comments ... 48
    Component descriptions ... 49
    Display style ... 49
Project documentation ... 49

## Chapter 2

**Define a controller component**

Introduction ... 51
Controller component ... 51
    L5X controller structure ... 51
    L5K CONTROLLER structure ... 52
    Controller elements ... 53
    Controller attributes ... 53
Controller attributes in a safety controller system ... 55
Controller guidelines ... 56
Examples ... 57

## Chapter 3

### Define a Datatype component

Introduction ... 65
Datatype component ... 65
- L5X datatype structure ... 65
- L5K datatype structure ... 65
- Datatype elements ... 65
- Datatype attributes ... 66

Datatype member ... 66
- L5X datatype member structure ... 66
- L5K datatype member structure ... 66
- Datatype member elements ... 67
- Datatype member attributes ... 67
- Bit members ... 67

Datatype guidelines ... 69
Examples ... 70

## Chapter 4

### Define a module component

Introduction ... 71
Module component ... 71
- L5X module structure ... 71
- L5K MODULE structure ... 72
- Module elements ... 72
- Module attributes ... 72

Module attributes in a safety controller system ... 76
Module connection ... 76
- L5X connection structure ... 76
- L5K CONNECTION structure ... 76
- Connection elements ... 77
- Module connection attributes ... 77
- Module connection attributes in a safety controller system ... 78

Module guidelines ... 79
Examples ... 80

## Chapter 5

### Define an Add-On Instruction component

Introduction ... 87
Add-On Instruction component ... 87
- L5X AddOnInstructionDefinition Structure ... 87
- L5K ADD_ON_INSTRUCTION_DEFINITION structure ... 88
- Add-On Instruction elements ... 88
- Add-On Instruction attributes ... 88
- Routines in Add-On Instructions ... 89

Parameters ... 90
- L5X parameters structure ... 90

L5K parameters structure ... 91
Parameter elements ... 91
Parameters attributes ... 93
Signature history ... 94
L5X SignatureHistory structure ... 94
L5K HISTORY_ENTRY structure ... 94
History entry attributes ... 94
Local tags ... 95
L5X LocalTags structure ... 95
L5K LOCAL_TAGS structure ... 96
Local tag attributes ... 96
Encoded/Unencoded Add-On Instructions ... 96
L5X EncodedData Structure ... 96
L5K ENCODED_DATA Structure ... 97
Encoded data attributes ... 97
Encoded Information elements ... 97
Encoded key attributes ... 97
Encoded content attributes ... 98
L5X Encoded Add-On Instruction example ... 99
L5K Encoded Add-On instruction example ... 100
Add-On Instruction Guidelines ... 100
L5X unencoded AddOnInstruction definition example ... 101
Add-On Instruction Example, Continued ... 102
L5K unencoded ADD_ON_INSTRUCTION _DEFINITION example 102
L5X unencoded safety AddOnInstruction definition example ... 104
L5K unencoded safety ADD_ON_INSTRUCTION _DEFINITION example ... 105

## Chapter 6

## Define a tag component

Introduction ... 107
Tag component ... 107
L5X tag structure ... 107
L5K TAG structure ... 108
Tag elements ... 108
Tag attributes ... 110
Produced tag attributes ... 111
Consumed tag attributes ... 112
ALARM_ANALOG tag ... 112
L5X tag structure for ALARM_ANALOG tag ... 112
L5K tag structure for ALARM_ANALOG tag ... 113
ALARM_ANALOG tag attributes ... 113
ALARM_DIGITAL tag ... 115
L5X tag structure for ALARM_DIGITAL tag ... 115
L5K tag structure for an ALARM_DIGITAL tag ... 115
ALARM_DIGITAL tag attributes ... 115

L5X AlarmConfig structure ..... 116
AlarmConfig elements ..... 117
L5K ANALOG_ALARM message structure ..... 117
ALMMSG elements ..... 117
MESSAGE tag ..... 118
L5X message structure ..... 118
L5K MESSAGE structure ..... 118
Message tag attributes ..... 118
AXIS_CIP_DRIVE, AXIS_CONSUMED, AXIS_GENERIC_DRIVE, AXIS_SERVO, AXIS_SERVO_DRIVE, or AXIS_VIRTUAL Tag ..... 119
L5X axis structure ..... 119
L5K AXIS TAG structure ..... 119
Axis tag attributes ..... 119
COORDINATE_SYSTEM tag ..... 131
L5X CoordinateSystem structure ..... 131
L5K COORDINATE_SYSTEM structure ..... 131
Coordinate system tag attributes ..... 132
MOTION_GROUP Tag ..... 133
L5X MotionGroup structure ..... 133
L5K MOTION_GROUP structure ..... 133
Motion Group Tag attributes ..... 133
HMIBC tag ..... 133
L5X HMIBC structure ..... 133
L5K HMIBC structure ..... 134
HMIBC attributes ..... 134
SAFETY tag ..... 134
L5X safety structure ..... 134
L5K SAFETY structure ..... 134
Safety tag attributes ..... 134
Tag initial values ..... 135
L5X initial tag value ..... 135
L5K initial TAG value ..... 135
Add-On Instruction tag values ..... 136
Add-On Instruction tag values example ..... 136
Tag guidelines ..... 137
Examples ..... 138

## Chapter 7

## Define a tag-based alarm or alarm definition

Introduction ..... 143
Tag-based alarms ..... 143
L5X Tag-based alarm structure ..... 143
L5K Tag-based alarm structure ..... 145
Tag-based alarm objects ..... 148
Tag-based alarm definitions ..... 152
L5X Tag-based alarm definition structure ..... 152

L5K Tag-based alarm definition structure ..... 153
Tag-based alarm definition objects ..... 154

## Chapter 8

### Define a program component

Introduction ..... 157
Program component ..... 157
L5X program structure ..... 157
L5K PROGRAM structure ..... 158
Program elements ..... 158
Program attributes ..... 158
Program attributes for EquipmentPhase programs ..... 159
Program attributes for Sequence programs ..... 160
Child program component ..... 160
L5K CHILD_PROGRAM structure ..... 161
L5X child program structure ..... 161
Child program attributes ..... 161
Child program guidelines ..... 161
Encoded/Unencoded routines ..... 161
L5X EncodedData Structure ..... 162
L5K ENCODED_DATA Structure ..... 162
Encoded Data Elements ..... 162
Encoded Information elements ..... 162
Encoded key attributes ..... 163
Encoded content attributes ..... 163
L5K source protected routine example ..... 163
L5X source-protected routine examples ..... 163
Program guidelines ..... 163
Examples ..... 164

## Chapter 9

### Define a ladder logic routine

Introduction ..... 167
Ladder logic routine ..... 167
L5X ladder logic routine structure ..... 167
L5K Ladder Logic ROUTINE structure ..... 167
Ladder logic routine elements ..... 167
RLL Routine attributes ..... 168
Rung logic ..... 168
L5X rung structure ..... 168
L5K RUNG structure ..... 168
Rung elements ..... 168
Rung attributes ..... 169
Rung guidelines ..... 169
Branches ..... 169
L5X branch structure ..... 169

L5K BRANCH structure....169
L5X Examples....170
L5K examples....170
Neutral text for ladder instructions....171

## Chapter 10

### Define a function block diagram routine

Introduction....179
Function BlockDiagram Routine....179
L5X function block diagram routine structure....179
L5K Function Block FBD_ROUTINE structure....179
Function block routine elements....179
FBD_routine attributes....180
Sheet....180
L5X sheet structure....180
L5K SHEET structure....181
Sheet elements....181
Sheet attributes....182
Sheet guidelines....182
Export function block logic while editing online....182
Input and output references....184
L5X IREFand OREF structure....184
L5K IREF and OREF structure....184
Reference attributes....184
Reference guidelines....184
Input and output connectors....185
L5X ICON and OCON structure....185
L5K ICON and OCON structure....185
Connector attributes....185
Connector guidelines....185
Blocks....185
L5X block structure....185
L5K BLOCK structure....185
Block attributes....186
Block guidelines....186
Functions....187
L5X function structure....187
L5K FUNCTION structure....187
Function attributes....187
Add-On instructions....187
L5X Add-On Instruction structure....187
L5K ADD_ON_INSTRUCTION structure....187
Add-On Instruction Attributes....187
Add-On Instruction Guidelines....188
JSR....188
L5X JSR structure....188

L5K JSR structure ..... 188
JSR attributes ..... 188
JSR guidelines ..... 189
SBR ..... 189
L5X SBR structure ..... 189
L5K SBR structure ..... 189
SBR attributes ..... 189
SBR guidelines ..... 189
RET ..... 190
L5X RET structure ..... 190
L5K RET structure ..... 190
RET attributes ..... 190
RET guidelines ..... 190
Wires and feedback wires ..... 190
L5X wire structure ..... 190
L5K WIRE structure ..... 190
Wire attributes ..... 190
Wire guidelines ..... 191
Text boxes ..... 191
L5X TextBox structure ..... 191
L5K TEXTBOX structure ..... 191
Text box attributes ..... 191
Text box guidelines ..... 192
Attachments ..... 192
L5X attachment structure ..... 192
L5K ATTACHMENT structure ..... 192
Attachment attributes ..... 192
Attachment guidelines ..... 192
Examples ..... 193
Parameters for function block instructions ..... 196

## Chapter 11

## Define a sequential function chart routine

Introduction ..... 201
Sequential function chart routine ..... 201
L5X sequential function chart structure ..... 201
L5K sequential function chart SFC_ROUTINE structure ..... 201
Sequential function chart elements ..... 202
SFC_Routine attributes ..... 202
Export sequential function chart logic while editing online ..... 203
Steps ..... 205
L5X step structure ..... 205
L5K STEP structure ..... 205
Step elements ..... 205
Step attributes ..... 206
Preset ..... 206

L5X preset structure ..... 206
L5K PRESET structure ..... 206
Limit high ..... 207
L5X limit high structure ..... 207
L5K LIMITHIGH structure ..... 207
Limit low ..... 207
L5X limit low structure ..... 207
L5K LIMITHIGH structure ..... 207
Action list ..... 207
L5X Action Structure ..... 207
L5K ACTION structure ..... 208
Action attributes ..... 208
Transitions ..... 208
L5X transition structure ..... 208
L5K TRANSITION structure ..... 208
Transition elements ..... 208
Transition attributes ..... 209
Condition ..... 209
L5X condition structure ..... 209
L5K CONDITION structure ..... 209
Subroutine calls ..... 209
L5X SbrRet structure ..... 209
L5K SBR_RET structure ..... 209
Subroutine attributes ..... 210
Stops ..... 210
L5X stop structure ..... 210
L5K STOP structure ..... 210
Stop attributes ..... 210
Branches ..... 210
L5X branch structure ..... 210
L5K BRANCH structure ..... 210
Branch attributes ..... 211
Leg attributes ..... 211
Directed links ..... 211
L5X DirectedLink structure ..... 211
L5K DIRECTED_LINK structure ..... 211
Directed link attributes ..... 211
Directed link guidelines ..... 211
Text boxes ..... 212
L5X TextBox structure ..... 212
L5K TEXTBOX structure ..... 212
Text box attributes ..... 212
Text box guidelines ..... 212
Attachments ..... 212
L5X attachment structure ..... 212
L5K ATTACHMENT structure ..... 212
Attachment attributes ..... 212

Attachment guidelines ..... 213
Examples ..... 213

## Chapter 12

### Define a structured text routine

Introduction ..... 221
Structured text routine ..... 221
L5X structured text structure ..... 221
L5K structured text ST_ROUTINE structure ..... 221
Structured Text routine elements ..... 221
ST_Routine attributes ..... 222
Structured text logic ..... 222
Export structured text logic while editing online ..... 223
Examples ..... 224
Structured text ..... 226

## Chapter 13

### Define an Equipment Sequence routine

Introduction ..... 233
Equipment Sequence Routine ..... 233
L5X Equipment Sequence structure ..... 233
L5K Equipment Sequence ESQ_ROUTINE structure ..... 234
Equipment Sequence elements ..... 234
ESQ_Routine attributes ..... 234
Steps ..... 235
L5X step structure ..... 235
L5K STEP structure ..... 235
Step attributes ..... 235
Transitions ..... 236
L5X transition structure ..... 236
L5K TRANSITION structure ..... 236
Transition elements ..... 236
Transition attributes ..... 237
Stops ..... 237
L5X stop structure ..... 237
L5K STOP structure ..... 237
Stop attributes ..... 237
Branches ..... 238
L5X branch structure ..... 238
L5K BRANCH structure ..... 238
Branch attributes ..... 238
Leg attributes ..... 238
Directed links ..... 238
L5X DirectedLink structure ..... 238
L5K DIRECTED_LINK structure ..... 238
Directed link attributes ..... 239

Directed link guidelines ..... 239
Attachments ..... 239
L5X attachment structure ..... 239
L5K ATTACHMENT structure ..... 239
Attachment attributes ..... 239
Attachment guidelines ..... 239
Tag configuration ..... 239
L5X tag configuration structure ..... 239
L5K TAG_CONFIGURATION structure ..... 240
Tag configuration elements ..... 240
Expression component ..... 240
L5X expression structure ..... 240
L5K EXPRESSION structure ..... 240

## Chapter 14

### Define a task component

Introduction ..... 241
Task component ..... 241
L5K TASK structure ..... 241
L5X task structure ..... 241
Task elements ..... 242
Task attributes ..... 242
Task guidelines ..... 243
Examples ..... 243

## Chapter 15

### Define a parameter connection component

Introduction ..... 245
Parameter connection component ..... 245
L5K PARAMETER_CONNECTION structure ..... 245
L5X ParameterConnection structure ..... 245
Parameter connection attributes ..... 245
Parameter connection guidelines ..... 245
Examples ..... 246

## Chapter 16

### Define a trend component

Introduction ..... 249
Trend component ..... 249
L5X trend structure ..... 249
L5K TREND structure ..... 249
Trend elements ..... 249
Trend attributes ..... 250
Pen declaration ..... 252
L5X pen structure ..... 252
L5K PEN structure ..... 252

Pen elements....253
Pen attributes....253
Trend guidelines....253
Examples....254

## Chapter 17

**Define a watch list component**

Introduction....255
Quick watch list component....255
L5X QuickWatchList structure....255
L5K QUICK_WATCH structure....255
Quick Watch elements....255
Quick Watch List attributes....255
Watch tag attributes....255
Examples....256

## Chapter 18

**Define controller configuration objects**

Introduction....257
Controller objects....257
L5X config structure....257
L5K CONFIG structure....257
Object elements....257
Config attributes....257
Examples....260

## Chapter 19

**Define custom properties**

Introduction....263
Custom properties data....263
Custom properties structure....263
Custom properties elements....264
Custom properties attributes....264
Example....264

## Chapter 20

**Structure Tags and Comments in an Import/Export File**

Introduction....265
Place information in a.CSV or.TXT file....265
Internal file comments....265
Specify a tag record....265
TAG type record....266
ALIAS type record....267
COMMENT type record....267
Specify a comment record....267

RCOMMENT Structure with Commas ..... 268
RCOMMENT Structure with Tabs ..... 268
TEXTBOX Structure with Commas ..... 268
TEXTBOX Structure with Tabs ..... 268
Specify an alarm message record ..... 269
ALMMSG Structure with Commas ..... 269
ALMMSG Structure with Tabs ..... 269
Examples ..... 270

## Appendix A

### Considerations for using Microsoft Excel to edit a.CSV file

Introduction ..... 275
Recommendations ..... 275
Logix Designer data transformations ..... 275
Microsoft Excel Data Transformation ..... 276

## Appendix B

### Import/Export revision history

Introduction ..... 277
Backward compatibility ..... 278
Import/Export version 2.22 Logix Designer application version 31 ..... 278
Import/Export version 2.21 Logix Designer application version 30 ..... 279
Import/Export version 2.20 Logix Designer application version 29 ..... 279
Import/Export version 2.19 Logix Designer application version 28 ..... 279
Import/Export version 2.18 Logix Designer application version 27 ..... 279
Import/Export version 2.17 Logix Designer application version 26 ..... 279
Import/Export version 2.15 Logix Designer application version 24 ..... 279
Import/Export version 2.12 Logix Designer application version 21 ..... 280
Import/Export version 2.11 Logix Designer version 20 ..... 281
Import/Export version 2.10 Logix Designer version 19 ..... 281
Import/Export version 2.8 Logix Designer version 18 ..... 281
Import/Export version 2.8 Logix Designer version 17 ..... 282
Import/Export version 2.7 Logix Designer version 16 ..... 282
Import/Export version 2.6 Logix Designer version 15 ..... 283
Import/Export version 2.4 Logix Designer version 13 ..... 284
Import/Export version 2.3 Logix Designer version 12 ..... 284
Import/Export version 2.2 Logix Designer version 11 ..... 285
Import/Export version 2.1 Logix Designer version 10 ..... 286
Changes to support MESSAGE tag enhancements ..... 286
Import/Export version 2.0 Logix Designer version 9 ..... 287
Motion changes to support the SERCOS Protocol ..... 288
MOTION_GROUP tag structure (version 1.1) ..... 288
AXIS tag structure (version 1.1) ..... 289
Import/Export version 1.1 Logix Designer version 8 ..... 291

## Index

This manual details how to import and export controller projects. You should be familiar with how the Logix-based controller stores and processes data. This manual is one of a set of related manuals that show common procedures for programming and operating Logix 5000 controllers.

For a complete list of common procedures manuals, refer to the Logix 5000 Controllers Common Procedures Programming Manual, publication 1756-PM001.

The term Logix 5000 controller refers to any controller based on the Logix 5000 operating system.

## Studio 5000 environment

The Studio 5000 Automation Engineering & Design Environment® combines engineering and design elements into a common environment. The first element is the Studio 5000 Logix Designer® application. The Logix Designer application is the rebranding of RSLogix 5000® software and will continue to be the product to program Logix 5000™ controllers for discrete, process, batch, motion, safety, and drive-based solutions.

The Studio 5000® environment is the foundation for the future of Rockwell Automation® engineering design tools and capabilities. The Studio 5000 environment is the one place for design engineers to develop all elements of their control system.

## Supported controllers

The supported controllers list includes:

- 1756-L71
- 1756-L71S
- 1756-L72
- 1756-L72S
- 1756-L73
- 1756-L73S
- 1756-L74
- 1756-L75
- 1756-L81E
- 1756-L81ES
- 1756-L81S
- 1756-L82E
- 1756-L82ES
- 1756-L82S
- 1756-L83E
- 1756-L83ES
- 1756-L83S
- 1756-L84E
- 1756-L84ES
- 1756-L84S
- 1756-L85E
- 1756-L85ES
- 1756-L85S
- 1769-L16ER-BB1B
- 1769-L18ER-BB1B
- 1769-L18ERM-BB1B
- 1769-L19ERM-BB1B
- 1769-L24ER-QB1B
- 1769-L24ER-QBFC1B
- 1769-L27ERM-QBFC1B

- 1769-L30ER
- 1769-L30ERM
- 1769-L30ERMS
- 1769-L30ER-NSE
- 1769-L33ER
- 1769-L33ERM
- 1769-L33ERMS
- 1769-L36ERM
- 1769-L36ERMS
- 1769-L37ERMO
- 1769-L37ERM
- 1769-L37ERMS
- 1769-L38ERM
- 1769-L38ERMS
- 1769-L37ERMOS
- 5069-L306ER
- 5069-L306ERM
- 5069-L306ERMS2
- 5069-L306ERMS3
- 5069-L306ERS2
- 5069-L310ER
- 5069-L310ERM
- 5069-L310ERMS2
- 5069-L310ERMS3
- 5069-L310ERS2
- 5069-L310ER-NSE
- 5069-L320ER
- 5069-L320ERM
- 5069-L320ERMS2
- 5069-L320ERMS3
- 5069-L320ERS2

- 5069-L330ER
- 5069-L330ERM
- 5069-L330ERMS2
- 5069-L330ERMS3
- 5069-L330ERS2
- 5069-L340ER
- 5069-L340ERM
- 5069-L340ERMS2
- 5069-L340ERMS3
- 5069-L340ERS2
- 5069-L350ERM
- 5069-L350ERMS2
- 5069-L350ERMS3
- 5069-L350ERS2
- 5069-L380ERM
- 5069-L380ERMS2
- 5069-L380ERMS3
- 5069-L380ERS2
- 5069-L3100ERM
- 5069-L3100ERMS2
- 5069-L3100ERMS3
- 5069-L3100ERS2
- 5069-L46ERMW
- Emulate 5570

## Additional resources

These documents contain additional information concerning related Rockwell Automation products.

| Resource | Description |
|---|---|
| Industrial Automation Wiring and Grounding Guidelines, publication 1770-4.1 | Provides general guidelines for installing a Rockwell Automation industrial system. |
| Product Certifications webpage, available at http://ab.rockwellautomation.com | Provides declarations of conformity, certificates, and other certification details. |

View or download publications at http://www.rockwellautomation.com/literature. To order paper copies of technical documentation, contact the local Rockwell Automation distributor or sales representative.

## Legal Notices

### Copyright notice

Copyright © 2018 Rockwell Automation Technologies, Inc. All Rights Reserved. Printed in USA.

This document and any accompanying Rockwell Software products are copyrighted by Rockwell Automation Technologies, Inc. Any reproduction and/or distribution without prior written consent from Rockwell Automation Technologies, Inc. is strictly prohibited. Please refer to the license agreement for details.

### End User License Agreement (EULA)

You can view the Rockwell Automation End-User License Agreement ("EULA") by opening the License.rtf file located in your product's install folder on your hard drive.

### Open source licenses

The software included in this product contains copyrighted software that is licensed under one or more open source licenses. Copies of those licenses are included with the software. Corresponding Source code for open source packages included in this product are located at their respective web site(s).

Alternately, obtain complete Corresponding Source code by contacting Rockwell Automation via the Contact form on the Rockwell Automation website: http://www.rockwellautomation.com/global/about-us/contact/contact.page Please include "Open Source" as part of the request text.

A full list of all open source software used in this product and their corresponding licenses can be found in the OPENSOURCE folder included with the Release Notes. The default installed location of these licenses is C:\Program Files (x86)\Common Files\Rockwell\Help\<*Product*>\ReleaseNotes\OPENSOURCE\index.htm.

### Trademark Notices

Allen-Bradley, ControlBus, ControlFLASH, Compact GuardLogix, Compact I/O, ControlLogix, CompactLogix, DCM, DH+, Data Highway Plus, DriveLogix, DPI, DriveTools, Explorer, FactoryTalk, FactoryTalk Administration Console, FactoryTalk Alarms and Events, FactoryTalk Batch, FactoryTalk Directory, FactoryTalk Security, FactoryTalk Services Platform, FactoryTalk

View, FactoryTalk View SE, FLEX Ex, FlexLogix, FLEX I/O, Guard I/O, High Performance Drive, Integrated Architecture, Kinetix, Logix5000, Logix 5000, Logix5550, MicroLogix, DeviceNet, EtherNet/IP, PLC-2, PLC-3, PLC-5, PanelBuilder, PowerFlex, PhaseManager, POINT I/O, PowerFlex, Rockwell Automation, RSBizWare, Rockwell Software, RSEmulate, Historian, RSFieldbus, RSLinx, RSLogix, RSNetWorx for DeviceNet, RSNetWorx for EtherNet/IP, RSMACC, RSView, RSView32, Rockwell Software Studio 5000 Automation Engineering & Design Environment, Studio 5000 View Designer, SCANport, SLC, SoftLogix, SMC Flex, Studio 5000, Ultra 100, Ultra 200, VersaView, WINtelligent, XM, SequenceManager are trademarks of Rockwell Automation, Inc.

Any Rockwell Automation logo, software or hardware product not mentioned herein is also a trademark, registered or otherwise, of Rockwell Automation, Inc.

**Other Trademarks**

CmFAS Assistant, CmDongle, CodeMeter, CodeMeter Control Center, and WIBU are trademarks of WIBU-SYSTEMS AG in the United States and/or other countries. Microsoft is a registered trademark of Microsoft Corporation in the United States and/or other countries. ControlNet is a trademark of ControlNet International. DeviceNet is a trademark of the Open DeviceNet Vendors Association (ODVA). Ethernet/IP is a trademark of ControlNet International under license by ODVA.

All other trademarks are the property of their respective holders and are hereby acknowledged.

**Warranty**

This product is warranted in accordance with the product license. The product's performance may be affected by system configuration, the application being performed, operator control, maintenance, and other related factors. Rockwell Automation is not responsible for these intervening factors. The instructions in this document do not cover all the details or variations in the equipment, procedure, or process described, nor do they provide directions for meeting every possible contingency during installation, operation, or maintenance. This product's implementation may vary among users.

This document is current as of the time of release of the product; however, the accompanying software may have changed since the release. Rockwell Automation, Inc. reserves the right to change any information contained in this document or the software at any time without prior notice. It is your responsibility to obtain the most current information available from Rockwell when installing or using this product.

### Environmental compliance

Rockwell Automation maintains current product environmental information on its website at http://www.rockwellautomation.com/rockwellautomation/about-us/sustainability-ethics/product-environmental-compliance.page

### Contact Rockwell Automation

Customer Support Telephone — 1.440.646.3434

Online Support — http://www.rockwellautomation.com/support/

# Import and export files

## Introduction

This document describes how to use the import/export feature that is included with the Logix Designer application.

With a Logix controller, you can import and export an entire project or import and export parts of a project. Select the import/export format based on what you want from the content.

| If you want: | Then use this format: |
|---|---|
| The entire controller project | L5K or L5X |
| Individual portions of the controller project | L5X |
| Tags and logic comments | CSV or TXT |

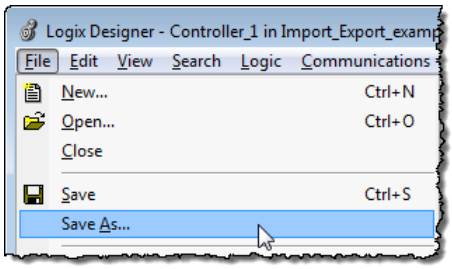
This chapter shows how to perform the import/export operations.

## Export a Project to an.L5K Text File

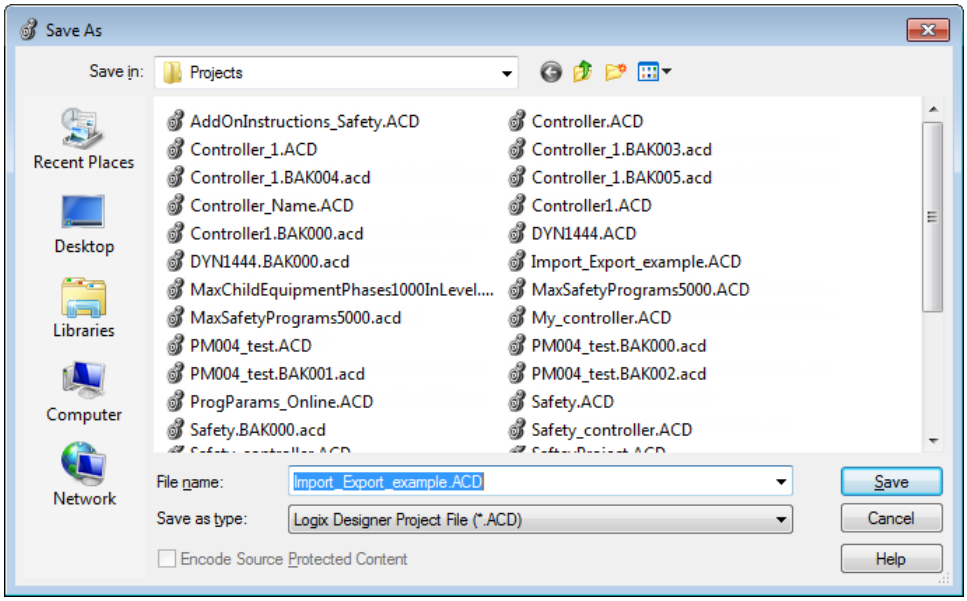
You can export a project to a text file and use any text editor that supports UTF8 file format to modify the project. The exported file will be an.L5K format.

Do these steps to export a project to an.L5K text file.

1. Make sure the project you want to export is open.
2. In the Logix Designer application, click **File > Save As**.



The **Save As** dialog box opens.




3. Browse to where you want to save the file.
4. In the **File name** field, type the name of the text file.
5. From the **Save as type** list, click the.L5K file format and click **Save**.

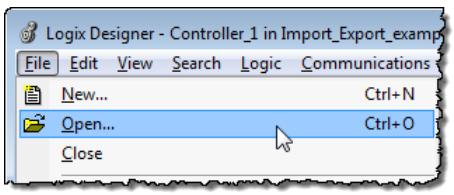
| **Important:** | The application automatically saves any unsaved edits when you click **OK**. |
|---|---|

## Import an.L5K text file

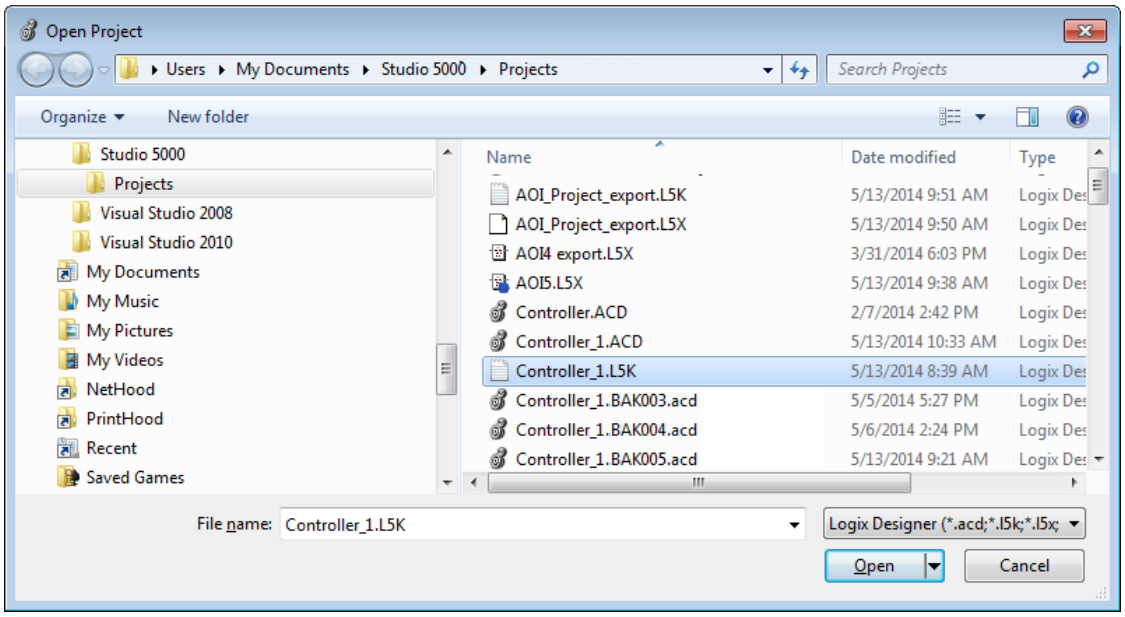
Import controller information from a saved text file that has an.L5K extension. This lets you use any text editor to create a project.

Do these steps to import an.L5K text file into a project.

1. In the Logix Designer application, from the **File** menu, choose **Open**.

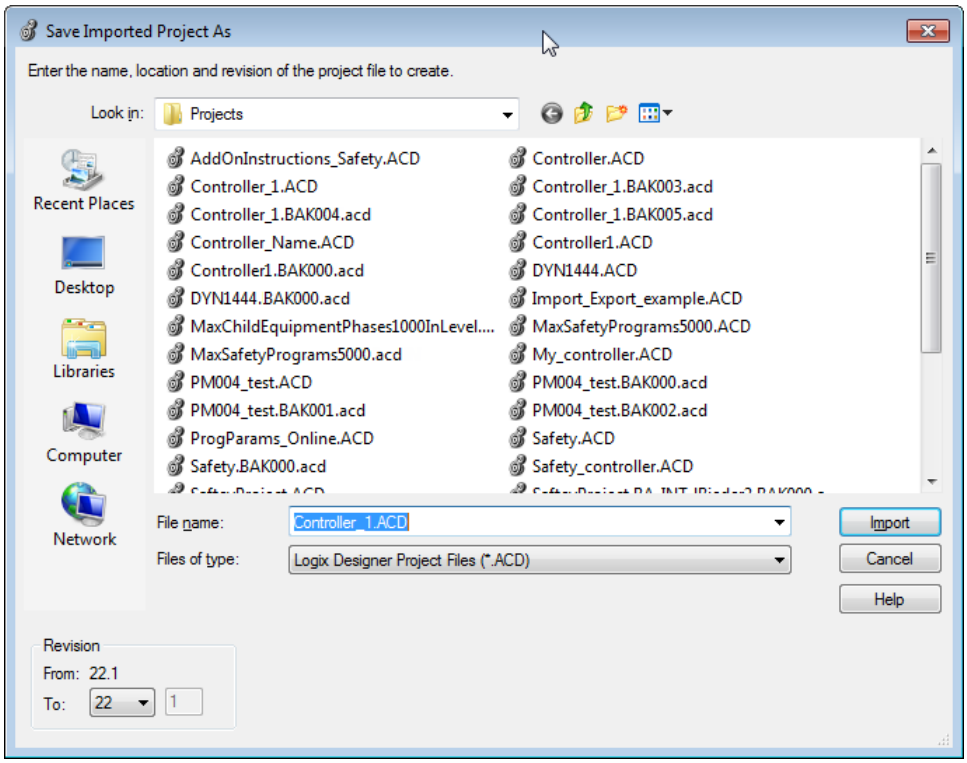


The **Open Import Project** dialog box opens.




2. Select the.L5K text file you want to import and click **Open**.

The **Save Imported Project As** dialog box opens.




3. Browse to where you want to save the imported project.

4. In the **File name** box, type the name for the imported project and click **Import**.

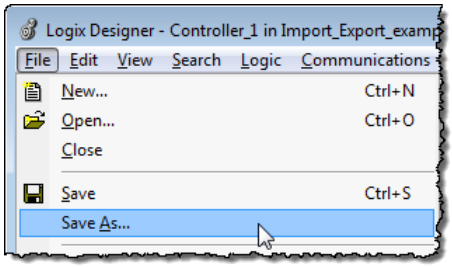
| | |
|---|---|
| **Important:** | If you import a project that has forces, the project defaults to **Forces Disabled**, even if the project was exported with **Forces Enabled**.<br><br>When you import an.L5K file, the project changes so that you cannot go online and access a previously downloaded controller. You must first upload from or download to the controller. See Maintaining Controller Access on page 37. |

## Export a Project to an.L5X XML File

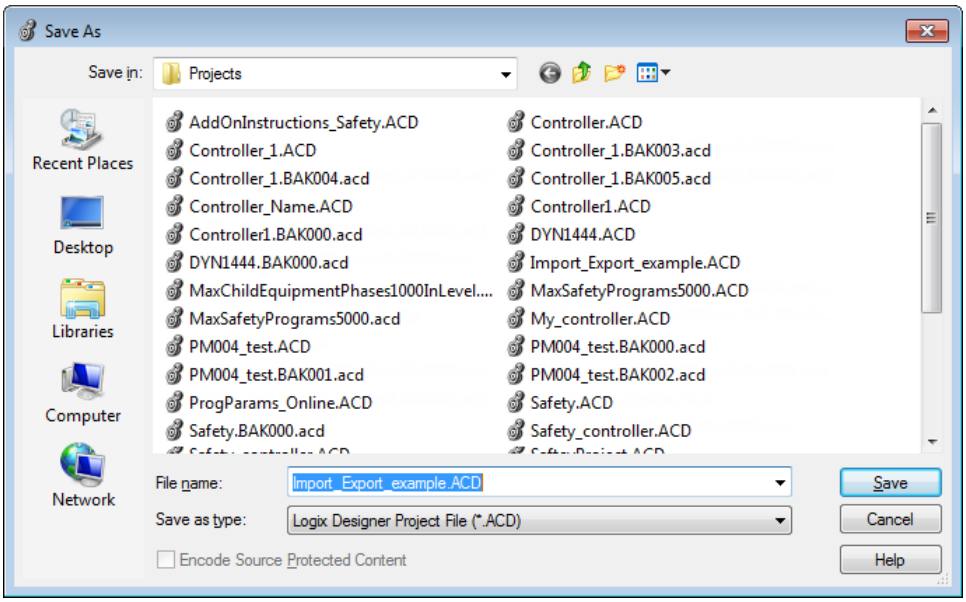
You can export a project to an XML file and use a text or XML editor to modify the project. The exported file will be an.L5X format.

Do these steps to export from a project to an.L5X XML file.

1. Make sure the project you want to export from is already open.
2. In the Logix Designer application, click **File > Save As**.




The **Save As** dialog box opens.




3. In the **File name** box, type the name of the file.

4. From the **Save as type** list, click.**L5X file format** and click **Save**.

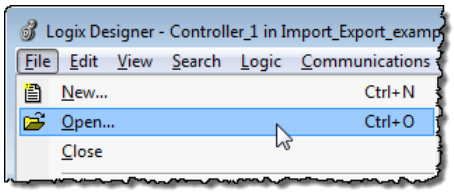
| **Important:** | The application automatically saves any unsaved edits when you click **OK**. |
|---|---|

## Import an.L5X XML File

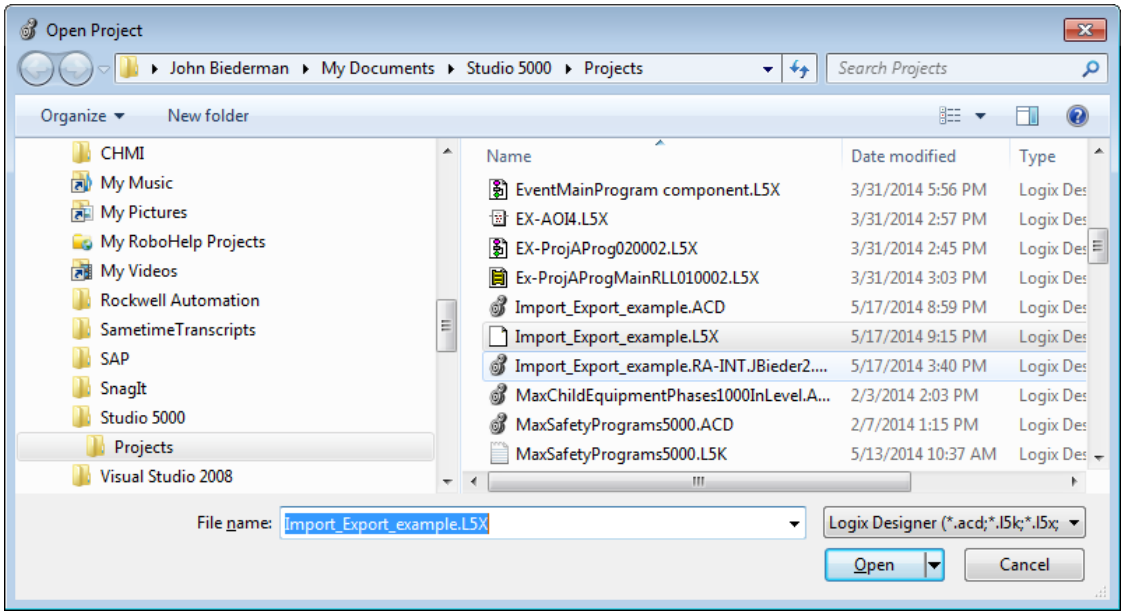
Import controller information from a saved XML file that has an.L5X extension and is a full controller export. This lets you use any editor to create a project.

Do these steps to import a controller.L5X XML file into a project.

1. In the Logix Designer application, click **File > Open**.

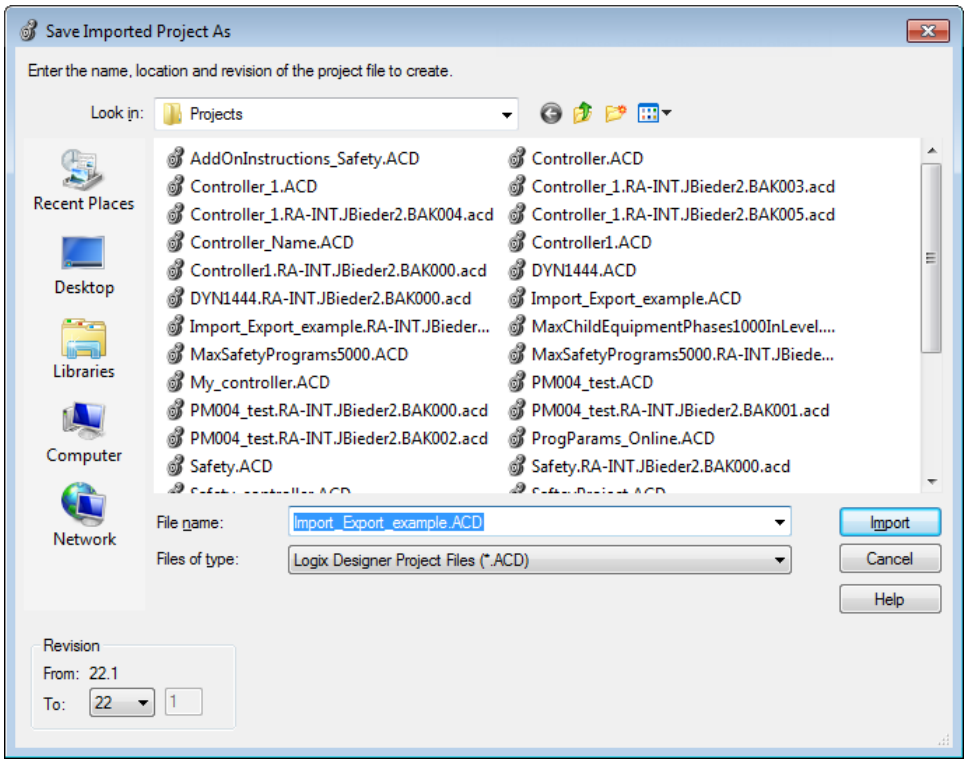



The **Open Project** dialog box opens.




2. In the **File name** box, select the.L5X controller file you want to import and click **Open**.

The **Save Imported Project As** dialog box opens.




3. Browse to where you want to save the imported project.
4. In the **File name** box, type the name for the project and click **Import**.

| | |
|---|---|
| **Important:** | If you import a project that has forces, the project defaults to **Forces Disabled**, even if the project was exported with **Forces Enabled**.<br><br>When you import an.L5X file, the project changes so that you cannot go online and access a previously downloaded controller. You must first upload from or download to the controller. See Maintaining Controller Access on page 37. |

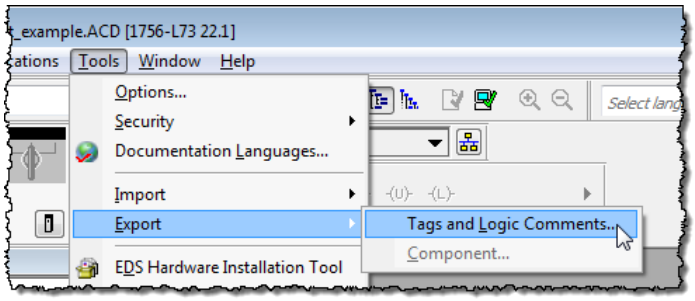
## Export to a.CSV or.TXT file

When you have a project open, you can export tags and logic comments to a structured file that separates values with commas (.CSV file) or that separates values with tabs (.TXT Unicode file). You can then use other applications, such as Microsoft Excel or Notepad, to edit the tags and logic comments.

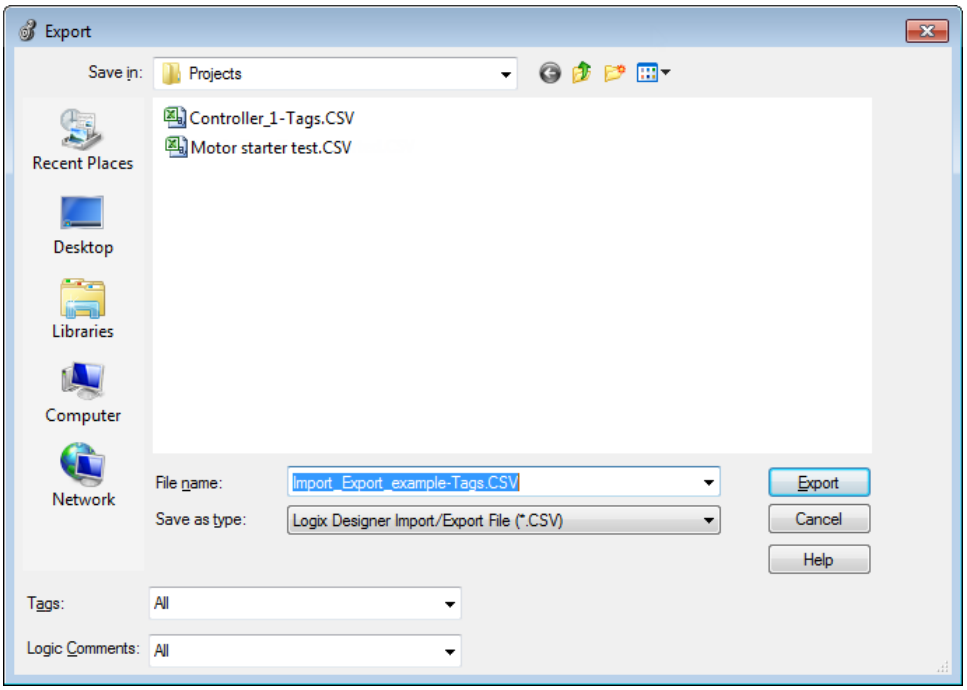
Do these steps to export tags and logic comments to a structured file.

1. Make sure the project from which you want to export tags and comments is already open.

2. In the Logix Designer application, click **Tools > Export > Tags and Logic Comments**.




The **Export** dialog box opens.

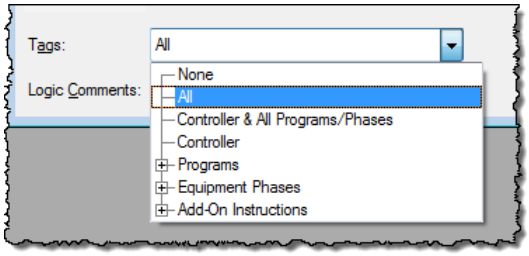



3. In the **File name** box, type the name of the file to be exported.

4. From the **Save as type** list, click.**CSV** or.**TXT** format.

   The.TXT import/export format supports double-byte characters, so you can use this format for all languages, including Chinese, Japanese, and Korean. The.CSV import/export format does not support double-byte characters.

5. From the **Tags and Logic Comments** list, set the scope of the tags and logic comments to be exported.




6. Click **Export**.

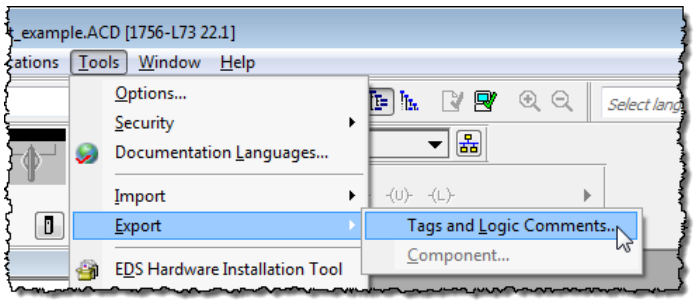
| Scope | Exported Material |
|---|---|
| All | All the tags (controller-scope, program-scope, equipment phase, and Add-On Instruction) or logic comments in the project |
| Controller and All Programs/Phases | Tags only; all controller-scope, program-scope, and equipment phase tags |
| Controller | Tags only; the controller-scoped tags of the project |
| All Programs/Phases | Logic Comments only; all program and equipment phase comments |
| Programs<br>Equipment Phases<br>Add-On Instructions | The tags or logic comments of a specific program, equipment phase, or Add-On Instruction |

## Import a.CSV or.TXT file

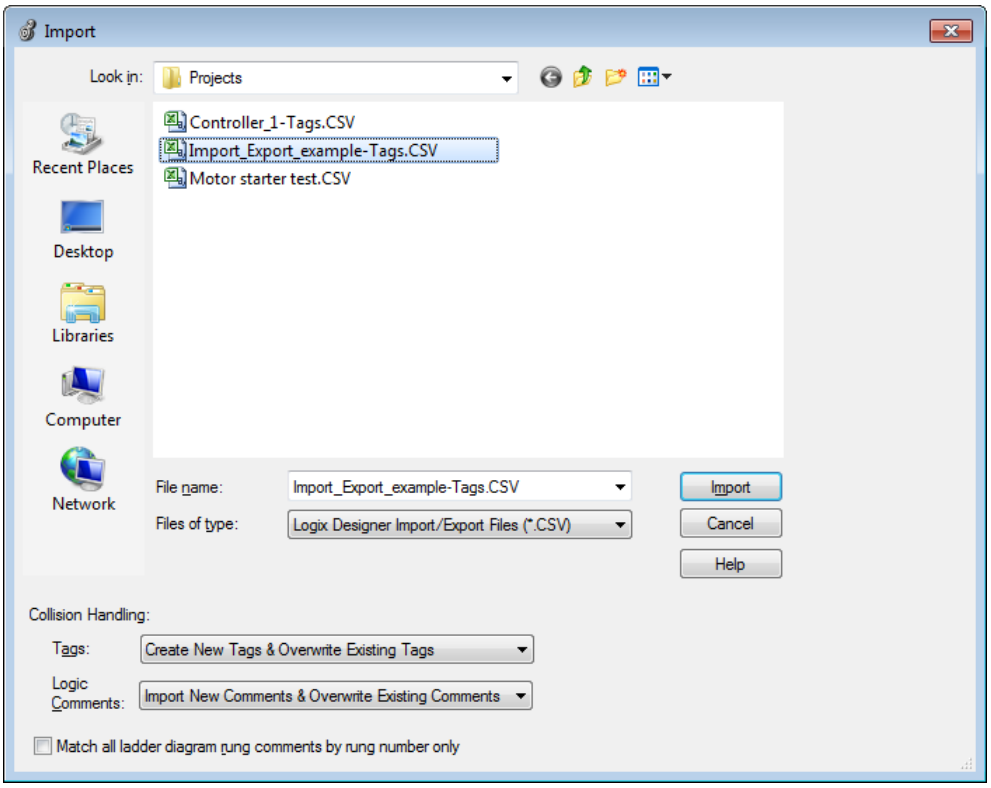
When you are offline and have a project open, you can import tags and logic comments from a saved.CSV file or.TXT file. This lets you use other applications, such as Microsoft Excel or Notepad, to create and edit tags and logic comments.

Do these steps to import tags and logic comments from a saved.CSV file or.TXT file into a project.

1. In the Logix Designer application, click **Tools > Import > Tags and Logic Comments**.



The **Import** dialog box opens.




2. In the **File name** box, select the.CSV or.TXT file you want to import.
3. From the **File of type** list, select.**CSV** or.**TXT** format.

   **Tip:** Use the **File of type** list to filter.CSV or.TXT files in the **Import** dialog box.

4. From the **Tags** lists, specify how you want to handle tag collisions.

| | |
|---|---|
| **Important:** | When you import tags, the tags in the import file may have the same name as tags that are already in the open project. This condition is a collision. |

| **If you want to:** | **From the Tags menu select:** |
|---|---|
| Replace tags in the project with tags from the import file, in addition to adding any new tags from the import file | **Create New Tags & Overwrite Existing Tags** |
| Keep tags that are in the project and discard colliding tags in the import file, in addition to adding any new tags from the import file | **Create New Tags & Overwrite Existing Tags** |
| Replace tags in the project with tags from the import file, but do not add any new tags from the import file | **Create New Tags & Overwrite Existing Tags** |

| | |
|---|---|
| **Important:** | If you delete tags from the.CSV or.TXT file and import the file, the process does not delete the tags from the controller project. Use the programming software to delete tags from the controller project. |

5. From the **Logic Comments** list, specify how you want to handle logic comment collisions.

| | |
|---|---|
| **Important:** | When you import logic comments, the possibility exists for the comments in the import file to differ from the comments in the open project when both are matched to the same logic. |

| If you want to: | From the Logic Comments list select: |
|---|---|
| Replace comments in the project with comments from the import file, in addition to adding any new comments from the import file | **Import New Comments & Overwrite Existing Comments** |
| Keep comments that are in the project and discard colliding comments in the import file, in addition to adding any new comments from the import file | **Import New Comments & Preserve Existing Comments** |
| Replace comments in the project with comments from the import file, but do not add any new comments from the import file | **Skip New Comments & Overwrite Existing Comments** |

6. Choose how to match comments to logic and click **Import**.

| If you want rung comments applied to: | Then: |
|---|---|
| The next rung that has the instruction, as specified in the Owning Element, as its last instruction on the rung | Make sure that the **Leave the Match all ladder diagram rung comments by rung only** check box is cleared.<br>This is the default and recommended option.<br>The Location element is ignored. |
| The rung number specified in the Location element | Select the **Match all ladder diagram rung comments by rung only** check box.<br>This overrides the default and recommended option.<br>The Owning Element is ignored. |
| **Important:** | If a.CSV file or.TXT file contains changes to tags, including aliases, when you import the file that the project changes such that you cannot go online and access a previously downloaded controller. You must first upload from or download to the controller.<br>If you only modify comments or descriptions before you import a.CSV file or.TXT file, you can go online with the controller. |

## Export source-protected logic

Starting with version 20, you can configure how source-protected content is exported in.L5K and.L5X files.

By default, source-protected content is now exported in an encrypted format to prevent viewing or modifying components in the system. A check box option on the **Workstation Options** dialog box enables Add-On Instructions and routines to be exported in a readable, cleartext format if the source keys for those components are present in the sk.dat file. This lets you modify protected content in a third-party tool, such as an XML editor.

| | |
|---|---|
| **Important:** | You must enable your workstation for source protection to use the cleartext option on the **Workstation Option** dialog box. Otherwise, the check box, is not available and source-protected content is exported in an encrypted format. |

Perform a partial or full-project export in cleartext when these parameters are available:

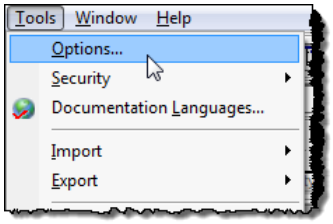
- Your workstation has source protection enabled.
- The **Workstation Option** dialog box (**Always Encode Source Protection Content On Export**) is not selected.
- The sk. dat file has been specified and it contains the source key (password) for the content. For details, see the Logix5000 Controllers Security Programming Manual , publication 1756-PM016.

If any of these requirements are absent, the content is exported in an encrypted format.

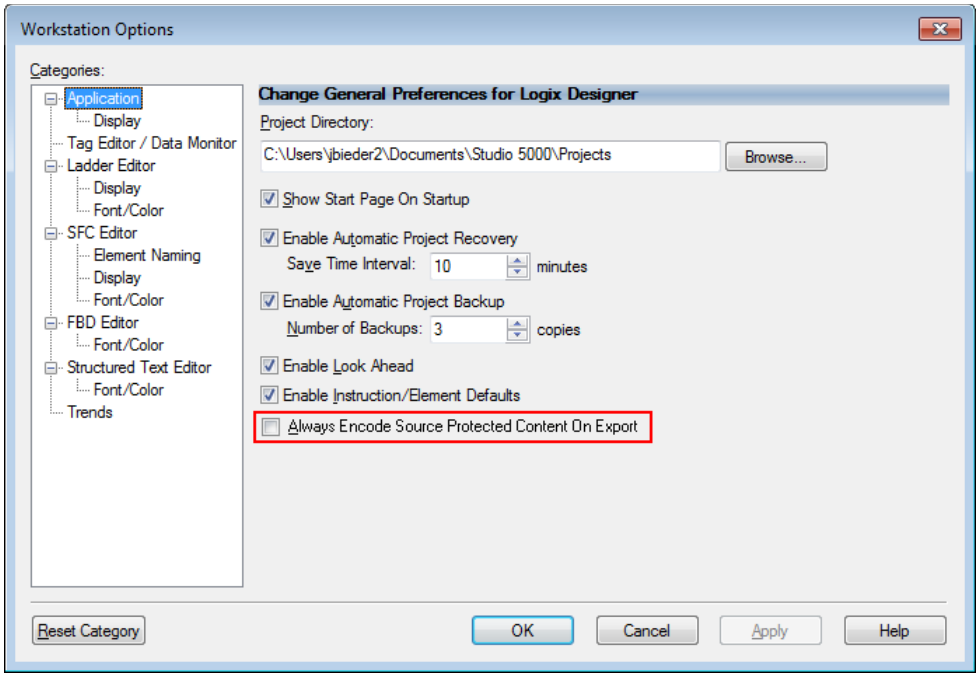
## Export in a Cleartext Format

Use the same procedure for partial exporting a component or a full project in readable text. Do these steps.

1. Open a project in the Logix Designer application that contains the content that you want to export.
2. On the **Menu** bar, click **Tools > Options**.




The **Workstation Options** dialog box opens with the export check box option **if** your personal computer is enabled for source protection.



3. Clear the **Always Encode Source Protected Content On Export** check box.

   There is a similar check box on the **Save As** and **Export** component dialog boxes that you must clear to export in cleartext from those dialog boxes.

   If you select the **Always Encode Source Protected Content On Export** check box on the **Workstation Options** dialog box, the application always exports source-protected content in an encrypted format, even when source keys for the content are present.

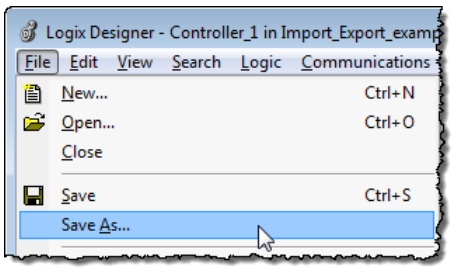
   | | |
   |---|---|
   | **Important:** | You cannot copy source-protected content from version 21 of the application and paste into earlier software versions. The pasting function is disabled in previous software versions when source-protected content is placed on the clipboard. |

4. Click **OK**.

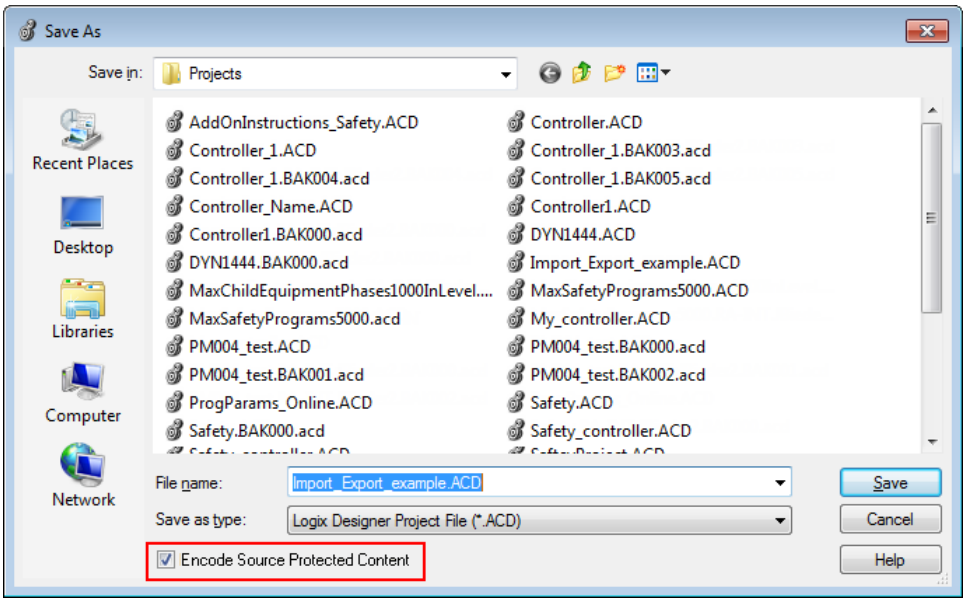
5. Take one of these actions:

   - To export a component, right-click the component and choose **Export Routine**. The component **Export** dialog box opens. Clear the **Encode Source Protected Content** check box. Click the **Export** button.
   - To save data for exporting as an.L5X or.L5K file, proceed to step 6.

6. On the **Menu** bar, click **File > Save As**.



The **Save As** dialog box opens.




7. From the **Save as type** list, select the.**L5K** or.**L5X** option, depending on your file type.

   The **Encode Source Protected Content** check box is available when you choose the.L5K or.L5X option.

| | |
|---|---|
| **Important:** | The **Encode Source Protected Content** check box is only enabled when the **Always Encode Source Protected Content on Export** check box in the **Workstation Options** dialog box is cleared.<br>The **Encode Source Protected Content** check box also appears on the component dialog box. |

8. Clear **Encode Source Protected Content** check box to export content in cleartext.

9. Click **Save**.

   See examples of encoded and unencoded codes for Add-On Instructions and Routines on page 96 and page 145, respectively.

## Maintaining controller access

The controller manages project status to provide Logix Designer application with the information to decide whether you can go online with a controller.

| Information | Description |
|---|---|
| Creation Stamp | The controller creates a creation stamp when you create, and import, a project and download the project to the controller. The creation stamp in the controller and the project file must match to enable Logix Designer application to go online with a controller.<br>If a project is exported to an.L5K file and then imported, the resulting project.ACD file gets a new creation stamp. This means that the Logix Designer application views the imported project as different from the file that was exported. The result is that you cannot use the new, imported project file to access a controller that was downloaded with the original file, before it was exported. At this point, your only options are to download again from the imported project file or to upload the controller contents to another project.ACD file and merge with the documentation from one of the older project.ACD files. |
| Download Stamp | The controller creates a download stamp on each download and stores this stamp in both the project and the controller. When the creation stamp and the download stamp in the controller match those in the project file, Logix Designer application can use the project to let you access the controller online.<br>If you change a project file when offline, that also clears the download stamp. This can occur when you import from an.L5X file or if you import a.CSV file that creates a new tag or modifies a tag data type. When you reset the download stamp, you can either download the project to the controller or upload the contents from the controller. If you choose to upload, you lose any changes made through import. Note that description and rung comment changes in a.CSV file do not reset the download stamp, so you can perform some.CSV file imports and still maintain access to the controller. |
| Change Log | Each time you change a controller online, the controller stores details about the change in a change log. If there are more than 1000 changes made to the project file since you last went online with the controller, you must either download the project to the controller or upload the contents from the controller. If you choose to upload, you lose any changes made through import. |

Given this status information, these situations prevent you from going online with a controller.

| Situations When You Cannot Go Online with a Controller | Possible Recovery |
|---|---|
| • More than 1000 controller edits were made.<br>• A download of another project copy with identical creation stamps occurred.<br>• Changes were made to the offline project, excluding documentation and tag value changes.<br>• A controller nonvolatile storage load occurred and the loaded image has the same creation stamp but a different download stamp.<br>• A controller nonvolatile storage load occurred and the loaded image has the same creation and download stamps but the change log is dated earlier than the project file. | • Full download to the controller<br>• Upload from the controller to a new project<br>• Upload from the controller and merge with an existing project. |
| • The project was exported and then reimported. In this case, the software considers it a different project and it has its own unique stamps.<br>• A different project, with different stamps, was downloaded.<br>• A controller nonvolatile storage load occurred, and the image was generated from a completely different project file, with different stamps. | • Full download to the controller<br>• Upload from the controller to a new project.<br><br>An upload/merge of documentation is not possible in these cases. |

## .L5X file structure

The.L5X import/export file consists of the components listed in the table. The.L5X format for each component is described in subsequent chapters of this reference manual.

| Component | Description | Chapter |
|---|---|---|
| <RSLogix5000Content> | Describes version and export information | This chapter, see example below |
| <Controller> | The controller | Chapter 2 Defining a Controller Component on page 51 |

| Component | Description | Chapter |
|---|---|---|
| <DataTypes> | User-defined and I/O data structures | Chapter 3 Defining a Datatype Component on page 65 |
| <Modules> | Modules in the controller organizer | Chapter 4 Defining a Module Component on page 71 |
| <AddOnInstructionDefinitions> | Add-On Instructions | Chapter 5 Defining an Add-On Instruction Component on page 87 |
| <Tags> | Controller-scope tags | Chapter 6 Defining a Tag Component on page 107 |
| <Programs> | Programs | Chapter 7 Defining a Program Component on page 157 |
| <Routines> | Ladder logic, function block diagram, sequential function chart, and structured text routines | Chapter 8 Defining a Ladder Logic Routine on page 167<br>Chapter 9 Defining a Function Block Diagram Routine on page 179<br>Chapter 10 Defining a Sequential Function Chart Routine on page 201<br>Chapter 11 Defining a Structured Text Routine on page 221 |
| <Tasks> | Controller tasks | Chapter 12 Defining a Task Component on page 241 |
| <ParameterConnection> | Parameter connections | Chapter 13 Defining a Parameter Connection on page 245 |
| <ParameterConnections> | Program parameter connections | Chapter 14 Defining a Parameter Connection on page 245 |
| <Trends> | Any trend configured for the controller project | Chapter 15 Defining a Trend Component on page 249 |
| <QuickWatchLists> | All quick watch lists configured in the controller project | Chapter 16 Defining a Watch List Component on page 255 |
| <CommPorts>, <CST>, and <WallClockTime> | Controller configuration objects | Chapter 17 Defining Controller Configuration Objects on page 257 |

The Controller component is the overall structure of the import/export file. It contains the configuration information and logic of the controller project. Preceding the Controller component is an optional XML declaration and a required root element tag (RSLogix5000Content) that includes Logix Designer application version information.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <RSLogix5000Content SchemaRevision="1.0" SoftwareRevision="26.00" TargetName="Controller_1"
    TargetType="Controller" ContainsContext="false" Owner="User, RA" ExportDate="Sun May 18
    13:02:10 2014" ExportOptions="DecoratedData ForceProtectedEncoding AllProjDocTrans">
```

All components in an.L5X import/export file follow this structure.

```
<component_type [attribute_list]>
    [body]
</component_type>
```

| Item | Description |
|---|---|
| <component_type> | The component begin tag. |
| attribute_list | List of attributes for the component in the form:<br>attribute_name="attribute_value"<br>Separate attributes with a space. |
| body | The content of the component. The content could include any subcomponents, for example, routines contained within a program, or a description of the component. The content could also include information about the component, for example, the data for a tag component.<br><br>The body of the component is optional. |

| Item | Description |
|---|---|
| Internet Explorer®</component_type>Internet ExplorerInternet Explorer | The component end tag.<br><br>A component with no content in the body may combine its begin and end tag as one tag:<br>(<component_type attribute_list /> |

The.L5X file is an ASCII file that is structured by using Extensible Markup Language (XML). In addition to being able to open and modify the.L5X file in a text editor, such as Notepad, you can also view the contents of the file in Internet Explorer® and other tools that work with.XML files.

| If you use: | You see: |
|---|---|
| A text editor, such as Notepad | A text file, such as:<br> <br>Edit this file in the text editor. |

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RSLogix5000Content SchemaRevision="1.0" SoftwareRevision="26.00" TargetName="My_controller"
TargetType="Controller" ContainsContext="false" Owner="User, RA" ExportDate="Mon May 19 08:26:42
2014" ExportOptions="DecoratedData ForceProtectedEncoding AllProjDocTrans">
<Controller Use="Target" Name="My_controller" ProcessorType="1756-L73" MajorRev="22"
MinorRev="1" TimeSlice="20" ShareUnusedTimeSlice="1" ProjectCreationDate="Wed Mar 26 13:55:49
2014" LastModifiedDate="Mon May 19 08:26:35 2014" SFCExecutionControl="CurrentActive"
SFCRestartPosition="MostRecent"
 SFCLastScan="DontScan" ProjectSN="16#0000_0000" MatchProjectToController="false"
CanUseRPIFromProducer="false" InhibitAutomaticFirmwareUpdate="0"
PassThroughConfiguration="EnabledWithAppend"
DownloadProjectDocumentationAndExtendedProperties="true">
<RedundancyInfo Enabled="false" KeepTestEditsOnSwitchOver="false" IOMemoryPadPercentage="90"
DataTablePadPercentage="50"/>
<Security Code="0" ChangesToDetect="16#ffff_ffff_ffff_ffff"/>
<SafetyInfo/>
<DataTypes/>
<Modules>
<Module Name="Local" CatalogNumber="1756-L73" Vendor="1" ProductType="14" ProductCode="94"
Major="22" Minor="1" ParentModule="Local" ParentModPortId="1" Inhibited="false"
MajorFault="true">
<EKey State="ExactMatch"/>
<Ports>
<Port Id="1" Address="0" Type="ICP" Upstream="false">
<Bus Size="7"/>
</Port>
</Ports>
</Module>
</Modules>
...
```

| If you use: | You see: |
|---|---|
| An Internet browser, such as Internet Explorer | An.XML file, such as:<br><?xml version="1.0" encoding="UTF-8" standalone="yes" ?><br>– <RSLogix5000Content SchemaRevision="1.0" SoftwareRevision="26.00" TargetName="My_controller" TargetType="Controller" ContainsContext="false" Owner="User, RA" ExportDate="Mon May 19 08:26:42 2014" ExportOptions="DecoratedData ForceProtectedEncoding AllProjDocTrans"><br>– <Controller Use="Target" Name="My_controller" ProcessorType="1756-L73" MajorRev="22" MinorRev="1" TimeSlice="20" ShareUnusedTimeSlice="1" ProjectCreationDate="Wed Mar 26 13:55:49 2014" LastModifiedDate="Mon May 19 08:26:35 2014" SFCExecutionControl="CurrentActive" SFCRestartPosition="MostRecent" SFCLastScan="DontScan" ProjectSN="16#0000_0000" MatchProjectToController="false" CanUseRPIFromProducer="false" InhibitAutomaticFirmwareUpdate="0" PassThroughConfiguration="EnabledWithAppend" DownloadProjectDocumentationAndExtendedProperties="true"><br><RedundancyInfo Enabled="false" KeepTestEditsOnSwitchOver="false" IOMemoryPadPercentage="90" DataTablePadPercentage="50" /><br><Security Code="0" ChangesToDetect="16#ffff_ffff_ffff_ffff" /><br><SafetyInfo /><br><DataTypes /><br>– <Modules><br>– <Module Name="Local" CatalogNumber="1756-L73" Vendor="1" ProductType="14" ProductCode="94" Major="22" Minor="1" ParentModule="Local" ParentModPortId="1" Inhibited="false" MajorFault="true"><br><EKey State="ExactMatch" /><br>– <Ports><br>– <Port Id="1" Address="0" Type="ICP" Upstream="false"><br><Bus Size="7" /><br></Port><br></Ports><br></Module><br></Modules><br>In the Internet browser, you can view only the file. Click the plus (+) and minus (-) to expand and collapse the viewable content. To edit the file, open the file in a text editor. |

## .L5X file conventions

The import/export feature L5X format is structured by using the Extensible Markup Language (XML). The XML specification is an open standard and is widely documented elsewhere. The only special convention used to describe the L5X format in this document is that items shown in square brackets ([ ]) are optional.

White space characters include spaces, tabs, carriage returns, new line, and form feeds. These characters can occur anywhere in an import/export file, except in keywords or names. If white space characters occur outside of descriptions, they are ignored.

## Internal file comments

Enter internal file comments to document your.L5X import files by using the XML commenting format. The import process ignores these comments. Place comments anywhere in an import/export.L5X file except within XML tag elements. You cannot next a comment within another comment.

An XML comment begins with the character sequence <!-- and ends with the character sequence -->. The comment appears between the begin and end character sequence. The character sequence -- may not appear within a comment. The text of the comment is ignored by the XML parser. This is an example.

```
<!-- This is a comment that includes an XML start tag element,
<mytag>. The start tag is ignored by the parser. -->
```

## Component Descriptions

Descriptions of components are optional. Unlike internal file comments, descriptions of components are imported. To add a description to a component, add a <Description> start tag element as the first sub-element of the component and then the description as a CDATA element in the body of the <Description> element.

Component descriptions are brought into the project without being processed by the XML parser for markup language. The description text is contained in a CDATA element, a standard in the XML specification. A CDATA element begins with the character sequence <![CDATA[ and ends with the character sequence ]]>. None of the text within the CDATA element is interpreted by the XML parser. The CDATA element preserves formatting so there is no need to use control characters to enter formatted descriptions.

```
<Task Name="Task1" Type="PERIODIC" Rate="1000" Priority="10"
Watchdog="500" DisableUpdateOutputs="false"
InhibitTask="false">
   <Description>
      <![CDATA[
         This is a task description with
         a line feed and " a quote.
      ]]>
   </Description>
</Task>
```

## Boolean attribute values

Many boolean attributes in the L5K format may have a value of 1 (enabled) or 0 (disabled). In the L5X format, these same attributes have a value of true (enabled) or false (disabled). Throughout this manual, a value of true or false should be specified in the L5X format for any attributes that indicate a value of 1 or 0.

## Data display style

Tags and data types support a radix attribute that specifies how to display the associated numerical information in the Logix Designer application.

| Radix Display Option | Example (based on 15 decimal) |
|---|---|
| Binary (uses a 2# prefix) | 2#0000_0000_0000_1111 |
| Octal (uses a 8# prefix) | 8#000_017 |
| Decimal | 15 |
| Hex (uses a 16# prefix) | 16#000F |
| Ascii | '$00$0F' |
| Exponential | 1.5000000e+01 |
| Float | 15.0 |

## Data formats

The Logix Designer application imports the data in tags, modules, and data structures (default values) with a <Data> element. You can import the <Data> element in three different formats from the.L5X file: raw, L5K, and decorated format. The format is indicated by a Format attribute.

```
<Tag Name="bINTtag" TagType="Base" DataType="INT"
   Radix="Decimal">
   <Data>05 00</Data>
   <Data Format="L5K">
      <![CDATA[5 ]]>
   </Data>
   <Data Format="Decorated">
      <DataValue DataType="INT" Radix="Decimal" Value="5"/>
   </Data>
</Tag>
```

The application exports all data values in both raw and decorated data format to the.L5X export file, with the exception that the InputTag under a module connection does not have the raw data format exported.

If multiple formats are present in the.L5X file, they must appear in the order of raw, L5K (if present), and decorated format. The data overwrites previous values if different. For example, if the decorated data is manipulated in the.L5X file, the decorated data values overwrite the raw data on import.

### *Raw data format*

Raw data format is a hex dump of the data and includes hidden (config) data. It is not intended to be readable. It is exported in the same format for all types of data types. The raw data format is the default format. A format attribute is not present if the <Data> element is in raw format.

| Data Type | Raw Data Export Format |
|---|---|
| Atomic | Example: an INT tag<br>`<Data>05 00</Data>` |
| Array of an Atomic Data Type | Example: an array of 4 INT tags<br>`<Data>02 00 03 00 04 00 05 00</Data>` |
| Structure Data Type | Example: a TIMER tag<br>`<Data>00 00 00 00 01 00 00 00 05 00 00 00</Data>` |
| Array of a Structure Data Type | Example: an array of 3 TIMER tags<br>`<Data>00 00 00 00 01 00 00 00 05 00 00 00 00 00 00 00 02 00 00 00 07 00 00 00 00 00 00 00 05 00 00 00 09 00 00 00</Data>` |

### *L5K Data Format*

L5K data format is the same data format that appears in an.L5K file, wrapped in a CDATA element. It is exported in the same format for all types of data types. It is indicated by the Format="L5K" attribute in the <Data> start tag element.

The L5K data format is not exported during a full project or component L5X format. However, it is imported if present in an.L5X file. This format is provided to enable customers to leverage the L5K data format as they convert their auto generation tools from L5K to L5X format.

| Data Type | L5K Data Export Format |
|---|---|
| Atomic | Example: an INT tag<br><Data Format="L5K"><br><![CDATA[5 ]]><br></Data> |
| Array of an Atomic Data Type | Example: an array of 4 INT tags<br><Data Format="L5K"><br><![CDATA[[2,3,4,5] ]]><br></Data> |
| Structure Data Type | Example: a TIMER tag<br><Data Format="L5K"><br><![CDATA[[0,1,5] ]]><br></Data> |
| Array of a Structure Data Type | Example: an array of 3 TIMER tags<br><Data Format="L5K"><br><![CDATA[[[0,1,5],[0,2,7],[0,5,9]] ]]><br></Data> |

### *Decorated Data Format*

Decorated data format is verbose and provides a human readable format of the data. It was added to the L5X format in version 17 of the application, for both L5X full project and L5X component exports and is recommended over the L5K format. The format of the data varies by data type. The decorated data format is indicated by the Format="Decorated" attribute in the <Data> start tag element.

| Data Type | Decorated Data Export Format |
|---|---|
| Atomic | Example: an INT tag<br><Data Format="Decorated"><br><DataValue DataType="INT" Radix="Decimal" Value="5"/><br></Data> |
| Array of an Atomic Data Type | Example: an array of 4 INT tags<br><Data Format="Decorated"><br><Array DataType="INT" Dimensions="4" Radix="Decimal"><br><Element Index="[0]" Value="2"/><br><Element Index="[1]" Value="3"/><br><Element Index="[2]" Value="4"/><br><Element Index="[3]" Value="5"/><br></Array><br></Data> |

| Data Type | Decorated Data Export Format |
|---|---|
| Structure Data Type | Example: a TIMER tag<br><Data Format="Decorated"><br><Structure DataType="TIMER"><br><DataValueMember Name="PRE" DataType="DINT" Radix="Decimal"<br>Value="1"/><br><DataValueMember Name="ACC" DataType="DINT" Radix="Decimal"<br>Value="5"/><br><DataValueMember Name="EN" DataType="BOOL" Value="0"/><br><DataValueMember Name="TT" DataType="BOOL" Value="0"/><br><DataValueMember Name="DN" DataType="BOOL" Value="0"/><br></Structure><br></Data> |

| Data Type | Decorated Data Export Format |
|---|---|
| Array Structure Data Type | Example: an array of 3 TIMER tags |

```
<Data Format="Decorated">
  <Array DataType="TIMER" Dimensions="3">
    <Element Index="[0]">
      <Structure DataType="TIMER">
        <DataValueMember Name="PRE" DataType="DINT" Radix="Decimal"
          Value="1"/>
        <DataValueMember Name="ACC" DataType="DINT" Radix="Decimal"
          Value="5"/>
        <DataValueMember Name="EN" DataType="BOOL" Value="0"/>
        <DataValueMember Name="TT" DataType="BOOL" Value="0"/>
        <DataValueMember Name="DN" DataType="BOOL" Value="0"/>
      </Structure>
    </Element>
    <Element Index="[1]">
      <Structure DataType="TIMER">
        <DataValueMember Name="PRE" DataType="DINT" Radix="Decimal"
         Value="2"/>
        <DataValueMember Name="ACC" DataType="DINT" Radix="Decimal"
         Value="7"/>
        <DataValueMember Name="EN" DataType="BOOL" Value="0"/>
        <DataValueMember Name="TT" DataType="BOOL" Value="0"/>
        <DataValueMember Name="DN" DataType="BOOL" Value="0"/>
      </Structure>
    </Element>
    <Element Index="[2]">
      <Structure DataType="TIMER">
        <DataValueMember Name="PRE" DataType="DINT" Radix="Decimal"
          Value="5"/>
        <DataValueMember Name="ACC" DataType="DINT" Radix="Decimal"
          Value="9"/>
        <DataValueMember Name="EN" DataType="BOOL" Value="0"/>
        <DataValueMember Name="TT" DataType="BOOL" Value="0"/>
        <DataValueMember Name="DN" DataType="BOOL" Value="0"/>
      </Structure>
    </Element>
  </Array>
</Data>
```

**Tip:** If the decorated data for a tag exceeds 100 KB, the decorated data format will not be exported. In this case, the raw data format will preserve the data values of the tag.

### *Force Values*

Force values are indicated with an additional <ForceData> XML tag element for both raw and L5K data formats. However, for decorated data, force values are indicated with ForceValue attribute in the <DataValue> XML tag element of the decorated data format.

```
<Tag Name="aDINTTag" TagType="Produced"
  DataType="DINT" Radix="Decimal">
  <ProduceInfo ProduceCount="1"
    ProgrammaticallySendEventTrigger="false"
    UnicastPermitted="false"/>
  <Data>03 00 00 00</Data>
  <ForceData>00 00 00 00 FF FF FF FF 07 00 00
    00
  </ForceData>
  <Data Format="L5K">
    <![CDATA[3 ]]>
  </Data>
  <ForceData Format="L5K">
    <![CDATA[[0,0,0,0,-1,-1,-1,-1,7,0,0,0] ]]>
  </ForceData>
  <Data Format="Decorated"><DataValue
    DataType="DINT" Radix="Decimal" Value="3"
    ForceValue="7"/>
  </Data>
</Tag>
```

## .L5K file structure

The.L5K import/export file contains these components. The L5K format for each component is described in subsequent chapters of this reference manual.

| Component | Description | See Chapter |
|---|---|---|
| CONTROLLER | The controller | Chapter 2 Define a Controller Component on page 51 |
| DATATYPE | User-defined and I/O data structures | Chapter 3 Define a Datatype Component on page 65 |
| MODULE | Modules in the controller organizer | Chapter 4 Define a Module Component on page 71 |
| ADD_ON_INSTRUCTION_ DEFINITION | Add-On Instructions | Chapter 5 Define an Add-On Instruction Component on page 87 |
| TAG | Controller-scope tags | Chapter 6 Define a Tag Component on page 107 |
| PROGRAM | Program files | Chapter 7 Define a Program Component on page 157 |
| ROUTINE | Ladder logic routines | Chapter 8 Define a Ladder Logic Routine on page 167 |
| FBD_ROUTINE | Function block diagram routines | Chapter 9 Define a Function Block Diagram Routine on page 179 |
| SFC_ROUTINE | Sequential function chart routine | Chapter 10 Define a Sequential Function Chart Routine on page 201 |
| ST_ROUTINE | Structured text routine | Chapter 11 Define a Structured Text Routine on page 221 |
| TASK | Controller tasks | Chapter 12 Define a Task Component on page 241 |
| PARAMETER_CONNECTION | Program Parameter connections | Chapter 13 Define a Parameter Connection on page 245 |
| TREND | Any trend configured for the controller project | Chapter 14 Define a Trend Component on page 249 |
| WATCH_LIST | All quick watch lists configured in the controller project | Chapter 15 Define a Watch List Component on page 255 |
| CONFIG | Configuration information | Chapter 16 Define Controller Configuration Objects on page 257 |

The CONTROLLER component is the overall structure of the import/export file. It contains the configuration information and logic of the controller project.

The header remarks (optional) and the version statement come before the CONTROLLER component.

```
Import-Export
Version  := RSLogix 5000 16.00
Owner    := User Name, Rockwell Automation
Exported := Tue May 13 08:39:40 2014
IE_VER := 2.20;
```

All components in an L5K import/export file follow this structure.

```
Component_Type <component_name> [Attributes]
   [body]
END_Component_Type
```

| Item | Description |
|---|---|
| Component_Type | The component. |
| component_name | A specific instance of the component. |
| Attributes | Any attributes of the component.<br>Component descriptions appear as an attribute of the component.<br>Separate each attribute with a comma (,). |
| body | Any subcomponents (children) of this component. |
| END_Component_Type | End of the component information. |

## .L5K file conventions

The import/export feature L5K format described in this document is based on the formats specified by the IEC 1131-3 specification.

| Convention | Description |
|---|---|
| < > | Items shown in angle brackets are required. |
| [ ] | Items shown in square brackets are optional. |
| user_value | Items in italics indicate user-supplied information. |
| LITERAL | Items in all uppercase indicate a required keyword or symbol that must be entered as shown. |
| "[" | Items in double quotes are required characters. |

White space characters include spaces, tabs, carriage returns, new line, and form feeds. These characters can occur anywhere in an import/export file, except in keywords or names. If white space characters occur outside of descriptions, they are ignored.

## Internal file comments

Enter comments to document import files. The import process ignores these comments. Place comments anywhere in an import/export file, except in names and descriptions. Enter comments by starting the line (record) with REMARK and a comma.

## Component descriptions

Descriptions of components are optional. Unlike internal file comments, descriptions are imported. The description of a component is added as an attribute of the component in the L5K format. Place the description within double quotes. See this example.

```
TASK Task1 (Description := "Hello World",
   Type := PERIODIC, Rate := 1000,
   Priority := 10, Watchdog := 500,
   DisableUpdateOutputs := No)
END_TASK
```

To enter control characters in the description, precede the character with a dollar sign ($). This table shows how to enter the supported control characters in a description.

| Character | Required Entry |
|---|---|
| $ | $$ |
| ‘ | $’ |
| " | $Q |
| 10 (line feed) | $L or $l |
| 13,10 (carriage return, line feed) | $N or $n |
| 12 (form feed) | $P or $p |
| 13 (carriage return) | $R or $r |
| 9 (tab) | $T or $t |
| xxxx (4-digit character code that represents a hexadecimal value) | $xxxx |

## Display style

Tags and data types support a radix attribute that specifies how to display the associated numerical information in the Logix Designer application.

| Radix Display Option | Example (based on 15 decimal) |
|---|---|
| Binary (uses a 2# prefix) | 2#0000_0000_0000_1111 |
| Octal (uses a 8# prefix) | 8#000_017 |
| Decimal | 15 |
| Hex (uses a 16# prefix) | 16#000F |
| Ascii | ‘$00$0F’ |
| Exponential | 1.5000000e+01 |
| Float | 15.0 |

## Project documentation

Starting with version 17 of the application, you can display project documentation, such as tag descriptions and rung comments for any supported localized language. You can store project documentation for multiple languages in

a single project file rather than in language-specific project files. Define all the localized languages that the project will support and set the current, default, and optional custom localized language. The software uses the default language if the content of the current language is blank for a particular component of the project. However, you can use a custom language to tailor documentation to a specific type of project file user.

Enter the localized descriptions in your Logix Designer application project when programming in that language or when using the import/export utility to translate the documentation off-line and import it back into the project. Once you enable project documentation in the Logix Designer application, you can dynamically switch between languages as you use the software.

These project documentation variables are available for any supported localized language:

- Component descriptions in tags, routines, programs, equipment phases, user-defined data types, and Add-On Instructions
- Engineering units and state identifiers added to tags, user-defined data types, or Add-On Instructions
- Trends
- Controllers
- Alarm messages (in configuration of ALARM_ANALOG and ALARM_DIGITAL tags)
- Tasks
- Property descriptions for a module in the Controller Organizer
- Rung comments, Sequential Function Chart text boxes, and Function Block Diagram text boxes

The localized comments are exported in all formats, L5K, L5X, CSV, and TXT. For more information on enabling a project to support multiple translations of project documentation, see the online help.

# Define a controller component

## Introduction

This chapter explains the overall structure of the controller component.

## Controller component

The controller component contains the overall structure of a project to be executed on one controller. It contains the configuration information and logic that you download to one controller.

### L5X controller structure

```
<Controller [controller_attributes]>
      <Description>
            <![CDATA[ text ]]>
      </Description>
      <RedundancyInfo [redundancyinfo_attributes]/>
      <Security [security_attributes]/>
            <PrimaryActionSets>
                  <PrimaryActionSet
PermissionSet=”PermissionSetName1” IsPermissionSet=”true”>
                        <![CDATA[ encoded_cached_permissions
]]>
                  </PrimaryActionSet>
                  <PrimaryActionSet
PermissionSet=”PermissionSetName2” IsPermissionSet=”false”>
                        <![CDATA[ encoded_cached_permissions
]]>
                  </PrimaryActionSet>
                  …
            </ PrimaryActionSets >
      </Security>
      <SafetyInfo [safetyinfo_attributes]>
            <SafetyTagMap>
            [comma separated safety tag map]
            </SafetyTagMap>
      </SafetyInfo>
      <DataTypes>
            [datatype]
      </DataTypes>
      <Modules>
            [module]
      </Modules>
      <AddOnInstructionDefinitions>
            [addoninstructiondefinition]
      </AddOnInstructionDefinitions>
      <Tags>
```

```
                [tag]
        </Tags>
        <Programs>
                [program]
        </Programs>
        <Tasks>
                [task]
        </Tasks>
        <ParameterConnections>
                [parameter_connection]
        </ParameterConnections>
        <CommPorts>
                [commport]
        </CommPorts>
        <CST [attribute_list] />
        <WallClockTime [attribute_list] />
        <Trends>
                [trend]
        </Trends>
        <QuickWatchLists>s
                [watch_list]
        </QuickWatchLists>
        < InternetProtocol  [Internet_Protocol_Attributes] />
<EthernetPorts>
        <EthernetPort [Ethernet_Port_Attributes] />
        <EthernetPort [Ethernet_Port_Attributes] />
</EthernetPorts>
<EthernetNetwork [Ethernet_Network_Attributes] />
</Controller>
```

**Tip:** The L5X controller structure must be contained within an RSLogix5000Content root element begin tag (<RSLogix5000Content>) and end tag (</RSLogix5000Content>). See the .L5X File Structure on page 38 in Chapter 1 for more information.

## L5K CONTROLLER structure

```
CONTROLLER <controller_name>  [(Description := "text",
                Controller_Attributes)]
        [<DATATYPE declaration>]
        [<MODULE declaration>]
        [<ADD_ON_INSTRUCTION_DEFINITION declaration]
        TAG
                [<tag declarations>]
        END_TAG
        [<PROGRAM declaration>]
        [<TASK declaration>]
        [PARAMETER_CONNECTION declaration]
        [<TREND declaration>]
        [<QUICK_WATCH declaration>]
        PRIMARY_ACTION_SET <permission_set_name>
(IsPermissionSet := Yes) :=
```

```
            encoded_cached_permissions;
        END_PRIMARY_ACTION_SET
        [<CONFIG controller objects declaration>]
END_CONTROLLER
```

## Controller elements

The table describes the elements comprising a controller.

| L5X Item | L5K Item | Description |
|---|---|---|
| N/A | *controller_name* | The name of the controller project.<br>In L5X, use a name attribute on the <Controller> element. |
| Description | Description | User information about the controller project |
| RedundancyInfo | N/A | For L5X format, this element contains redundancy attributes. |
| Security | N/A | For L5X format, this element contains security attributes. |
| SafetyInfo | N/A | For L5X format, this element contains safety controller attributes. |
| DataTypes | DATATYPE | I/O and user-defined data structures |
| Modules | MODULE | Devices in the controller organizer |
| AddOnInstructionDefinitions | ADD_ON_INSTRUCTION_<br>DEFINITION | Add-On instructions |
| Tags | TAG | Controller-scope tags |
| Programs | PROGRAM | Programs |
| Tasks | TASK | Tasks |
| ParameterConnections | PARAMETER_CONNECTIONS | Program parameter connections |
| Trends | TREND | Controller trends |
| QuickWatchLists | QUICK_WATCH | List of watch tags specified for a quick watch list |
| CommPorts, CST, WallClockTime | CONFIG | Characteristics of controller objects (status information) |
| PrimaryActionSet | PRIMARY_ACTION_SET | Cache of permissions for the Guest Users group that are associated with the specified logical name or permission set. |

## Controller attributes

| Attribute | Description |
|---|---|
| Use | L5X only. Specify context or target. |
| Name | L5X only. Specify the name of the controller component.<br><br>In L5K, the name is an element of the controller component. |
| ProcessorType | Specify the type of controller. (1756-L71, 1756-L71S, 1756-L72, 1756-L72S, 1756-L73, 1756-L73S, 1756-L74, 1756-L75, 1769-L16ER-BB1B, 1769-L18ER-BB1B, 1769-L18ERM-BB1B, 1769-L24ER-QB1B, 1769-L24ER-QBFC1B, 1769-L27ERM-QBFC1B, 1769-L30ER, 1769-L30ERM, 1769-L30ER-NSE, 1769-L33ER, 1769-L33ERM, 1769-L36ERM, Emulator) |
| Major | L5K only. Specify the major revision number (1...127) of the controller. |
| MajorRev | L5X only. Specify the major revision number (1...127) of the controller. |
| MinorRev | L5X only. Specify the minor revision number (1...127) of the controller. |

| Attribute | Description |
|---|---|
| TimeSlice | Percentage of available CPU time (10...90) that is assigned to communication. |
| ShareUnusedTimeSlice | Specify whether to share an unused timeslice or not. Type a **0** to not share; type a **1** to share. |
| PowerLossProgram | Name of the program to be executed upon restart after a power loss. |
| MajorFaultProgram | Name of the program to be executed when a major fault occurs. |
| CommPath | Specify the devices in the communication path. The communication path ends with the controller (\Backplane\1). This is exported only if you select manual configuration of the communication path in RSLinx software. |
| CommDriver | Specify the type of communication driver. This is the name of the selected driver in RSLinx software. This is exported only if you select manual configuration of the communication driver in RSLinx software. |
| RedundancyEnabled | L5K only. Specify whether redundancy is used or not. Type a **0** to disable redundancy; type a **1** to enable redundancy. |
| Enabled | L5X only. Specify whether redundancy is used (true or false).<br><br>This attribute is on the <RedundancyInfo> tag element. |
| KeepTestEditsOnSwitchOver | Specify whether to keep test edits on when a switchover occurs in a redundant system. Type a **0** not to keep test edits on; type a **1** to keep test edits on.<br><br>For L5X, this attribute is on the <RedundancyInfo> tag element. Type **false** or **true**. |
| IOMemoryPadPercentage | Specify the percentage (0...100) of I/O memory that is available to the system after the download when configured for redundancy.<br><br>For L5X, this attribute is on the <RedundancyInfo> tag element. |
| DataTablePadPercentage | Specify the percentage (0...100) of the data table to reserve. If redundancy is not enabled, type **0**. If redundancy is enabled, type **50**.<br><br>For L5X, this attribute is on the <RedundancyInfo> tag element. |
| SecurityCode | L5K only. Specify whether the RSI Security Server is enabled for the controller. Type **0** if the controller is unsecured; type a 10-digit, non-zero value if the controller is secured. |
| Code | L5X only. Specify whether the RSI Security Server is enabled for the controller. Type **0** if the controller is unsecured; type a 10-digit, non-zero value if the controller is secured.<br><br>This attribute is on the <Security> tag element. |
| SFCExecutionControl | Specify whether the SFC executes the current active steps before returning control (CurrentActive) or whether the SFC executes all threads until reaching a false transition (UntilFalse). |
| SFCRestartPosition | Specify whether the SFC restarts at the most recently executed step (MostRecent) or at the initial step (InitialStep). |
| SFCLastScan | Specify how the SFC manages its state on a last scan. Select AutomaticReset, ProgrammaticReset, or DontScan. |
| SerialNumber | L5K only. Specify the serial number of the controller. If a serial number is specified, it is imported into the project regardless of the MatchProjectToController setting. Type a 32-bit, hexadecimal number with the 16# prefix, such as 16#0012_E2BC. |
| ProjectSN | L5X only. Specify the serial number of the controller. If a serial number is specified, it is imported into the project regardless of the MatchProjectToController setting. Type a 32-bit, hexadecimal number with the 16# prefix, such as 16#0012_E2BC. |
| MatchProjectToController | Specify whether to be sure that the project matches the controller or not. Type **Yes** or **No**. |
| InhibitAutomaticFirmwareUpdate | Specify whether to inhibit the automatic update of controller firmware. Type a **0** to not inhibit; type a **1** to inhibit. |

| Attribute | Description |
|---|---|
| CurrentProjectLanguage | Specify the current project language for a project documentation project. |
| DefaultProjectLanguage | Specify the default project language for a project document at on project. |
| ControllerLanguage | Specify the controller project language for a project document at on project. |
| CanUseRPIFromController | Specify whether the consumed tags in the controller can connect to the producer with an RPI provided by the producer (true or false). |
| SecurityAuthorityID | ID of the FactoryTalk Diagnostics® to which your controller is bound.<br><br>For L5X only, this attribute is on the <Security> tag element. |
| SecurityAuthorityURI | Network path to the FactoryTalk Diagnostics to which your controller is bound.<br><br>For L5X only, this attribute is on the <Security> tag element. |
| PermissionSet | Name of the set of permissions, configured in FactoryTalk Security, to apply to this object.<br><br>For L5X only, this attribute is on the <Security> tag element. |
| IsPermissionSet | Indicates if this is associated with a permission set or a logical name. |
| ChangesToDetect | Mask that specifies the controller events that you wish to track.<br><br>For L5X only, this attribute is on the <Security> tag element. |
| TrustedSlots | Mask defining the slots through which the trusted communication is permitted to the controller.<br><br>For L5X only, this attribute is on the <Security> tag element |
| PassThroughConfiguration | For L5K and L5X. Indicates the pass through state of documentation for the project.<br>Type **Disabled**, **Enabled**, or **EnabledWithAppend** |
| DownloadProjectDocumentationAndExtendedProperties | For L5K and L5X. Indicates the download project documentation configuration setting of the project. |
| DownloadCustomProperties | For L5K and L5X. Indicates the download custom properties configuration setting of the project. Only applies if the project is already configured to DownloadProjectDocumentation.<br><br>Rockwell recommends setting this attribute to false only during startup testing to improve download speeds during commissioning testing. It should be set to true for the normal operating state of a system. For L5X, the setting is **true** or **false**. For L5K, the setting is **1** (true) or **0** (false). |
| EtherNetIPMode | The EtherNet/IP Mode describes the relationship between the CIP EtherNet/IP ports and the physical Ethernet ports. The CIP EtherNet/IP port can be configured as one of two modes:<br>• Dual-IP<br>• Linear/DLR |

## Controller attributes in a safety controller system

For safety controllers, specify these attributes for the Controller component, in addition to those previously described.

| Attribute | Description |
|---|---|
| SafetySignature | Specifies the safety signature control as defined in the controller properties. This value is exported only; it is ignored on import.<br><br>For L5X, this attribute is on the <SafetyInfo> tag element. |

| Attribute | Description |
|---|---|
| SafetyLocked | Displays whether the safety controller is locked or not. This value is exported only; it is ignored on import. This value will be Yes or No.<br><br>For L5X, this attribute is on the <SafetyInfo> tag element. Type **true** or **false**. |
| SafetyLockPassword | Specifies the lock password in the controller. This value is encrypted on export.<br><br>For L5X, this attribute is on the <SafetyInfo> tag element. |
| SafetyUnlockPassword | Specifies the unlock password in the controller. This value is encrypted on export.<br><br>For L5X, this attribute is on the <SafetyInfo> tag element. |
| SafetyTagMap | L5K only as an attribute. Specify the tags in the safety tag map. Place double quotes around the tags. Each entry must end with a comma and carriage return. This is an example.<br>"StdTag1=SafeTag1,<br>StdTag2=SafTag2"<br><br>For L5X, a <SafetyTagMap> element is a subelement under the <SafeyInfo> element. Specify the tags in the safety tag map in the body of the <SafetyTagMap> element Do not use quotes. Separate mappings with a comma and a space.<br><br>`- <SafetyInfo SafetyLocked="true"`<br>`SafetyLockPassword="VAAiAF4AMCChAKUABgBBABcA4ABaAJIBDAD2ALMAAgDGArsArgC6ACgA`<br>`SafetyUnlockPassword="TQAjAFEAjQDCAM4ABgBBABcA4ABaAJIBDAD2ALMAAgDGArsArgC6ACc`<br>`ConfigureSafetyIOAlways="false">`<br>`<SafetyTagMap>Standard_Tag=Safety_Tag, Other_Standard=Other_Safety</SafetyTagMap>`<br>`</SafetyInfo>` |
| ConfigureSafetyIOAlways | Specify whether to configure safety I/O when replacing safety I/O. Type **Yes** or **No**.<br><br>For L5X, this attribute is on the <SafetyInfo> tag element. Type **true** or **false**. |
| SignatureRunModeProtect | Indicates whether you can modify the safety signature when in Run mode.<br><br>For L5X only, this attribute is on the <SafetyInfo> tag element. |

## Controller guidelines

Observe these guidelines when defining a controller:

- All declarations must be ordered in the prescribed syntax.
- The maximum number of tasks vary by the controller type.

| Controller | Maximum Number of Tasks |
|---|---|
| ControlLogix® | 32 |
| SoftLogix™ 5800 | 32 |
| FlexLogix™ | 8 |
| CompactLogix™ | 4 |
| DriveLogix™ | 4 |

- There can be only one continuous task.
- Programs can be scheduled under only one task.

- There can be a maximum of 1000 programs under a task.
- Scheduled programs must be defined.

## Examples

**L5X Controller Example**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <RSLogix5000Content SchemaRevision="1.0" SoftwareRevision="26.00"
  TargetName="example_controller" TargetType="Controller" ContainsContext="false" Owner="User,
  RA" ExportDate="Mon May 19 10:13:11 2014" ExportOptions="DecoratedData
  ForceProtectedEncoding AllProjDocTrans">
  - <Controller Use="Target" Name="example_controller" ProcessorType="1756-L73" MajorRev="22"
    MinorRev="1" TimeSlice="20" ShareUnusedTimeSlice="1" ProjectCreationDate="Wed Mar 26
    13:55:49 2014" LastModifiedDate="Mon May 19 08:26:35 2014"
    SFCExecutionControl="CurrentActive" SFCRestartPosition="MostRecent" SFCLastScan="DontScan"
    ProjectSN="16#0000_0000" MatchProjectToController="false" CanUseRPIFromProducer="false"
    InhibitAutomaticFirmwareUpdate="0" PassThroughConfiguration="EnabledWithAppend"
    DownloadProjectDocumentationAndExtendedProperties="true">
    - <Description>
      <![CDATA[ controller description ]]>
    </Description>
    <RedundancyInfo Enabled="false" KeepTestEditsOnSwitchOver="false"
      IOMemoryPadPercentage="90" DataTablePadPercentage="50" />
    <Security Code="0" ChangesToDetect="16#ffff_ffff_ffff_ffff" />
    <SafetyInfo />
    <DataTypes />
  + <Modules>
  + <AddOnInstructionDefinitions>
  + <Tags>
  + <Programs>
  + <Tasks>
  + <ParameterConnections>
    <CST MasterID="0" />
    <WallClockTime LocalTimeAdjustment="0" TimeZone="0" />
  + <Trends>
    <DataLogs />
    <TimeSynchronize Priority1="128" Priority2="128" PTPEnable="false" />
  </Controller>
</RSLogix5000Content>
```

**L5X Safety Controller Example**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <RSLogix5000Content SchemaRevision="1.0" SoftwareRevision="26.00" TargetName="SafetyProject"
   TargetType="Controller" ContainsContext="false" Owner="User, RA" ExportDate="Mon May 19
   10:28:12 2014" ExportOptions="DecoratedData ForceProtectedEncoding AllProjDocTrans">
  - <Controller Use="Target" Name="SafetyProject" ProcessorType="1756-L73S" MajorRev="22"
     MinorRev="1" TimeSlice="20" ShareUnusedTimeSlice="1" ProjectCreationDate="Mon May 19
     10:22:08 2014" LastModifiedDate="Mon May 19 10:22:11 2014"
     SFCExecutionControl="CurrentActive" SFCRestartPosition="MostRecent" SFCLastScan="DontScan"
     ProjectSN="16#0000_0000" MatchProjectToController="false" CanUseRPIFromProducer="false"
     InhibitAutomaticFirmwareUpdate="0" PassThroughConfiguration="EnabledWithAppend"
     DownloadProjectDocumentationAndExtendedProperties="true">
    + <Description>
     <RedundancyInfo Enabled="false" KeepTestEditsOnSwitchOver="false"
       IOMemoryPadPercentage="90" DataTablePadPercentage="50" />
     <Security Code="0" ChangesToDetect="16#ffff_ffff_ffff_ffff" />
     <SafetyInfo SafetyLocked="true" SignatureRunModeProtect="false"
       SafetyLockPassword="VAAiAF4AMCChAKUABgBBABcA4ABaAJIBDAD2ALMAAgDGArsArgC6ACgAG
       SafetyUnlockPassword="TQAjAFEAjQDCAM4ABgBBABcA4ABaAJIBDAD2ALMAAgDGArsArgC6ACgA
       ConfigureSafetyIOAlways="false" >
       <SafetyTagMap>Standard_Tag=Safety_Tag,
         Other_Standard=Other_Safety</SafetyTagMap>
     </SafetyInfo>
     <DataTypes />
    + <Modules>
     <AddOnInstructionDefinitions />
    + <Tags>
    + <Programs>
    + <Tasks>
    + <ParameterConnections>
     <CST MasterID="0" />
     <WallClockTime LocalTimeAdjustment="0" TimeZone="0" />
     <Trends />
     <DataLogs />
     <TimeSynchronize Priority1="128" Priority2="128" PTPEnable="false" />
   </Controller>
</RSLogix5000Content>
```

**L5K CONTROLLER Example**

```
CONTROLLER example_controller (Description := "controller
description",
                ProcessorType := "1756-L73",
                Major := 22,
                TimeSlice := 20,
                ShareUnusedTimeSlice := 1,
                RedundancyEnabled := 0,
                KeepTestEditsOnSwitchOver := 0,
                DataTablePadPercentage := 50,
```

```
                    SecurityCode := 0,
                    ChangesToDetect := 16#ffff_ffff_ffff_ffff,
                    SFCExecutionControl := "CurrentActive",
                    SFCRestartPosition := "MostRecent",
                    SFCLastScan := "DontScan",
                    SerialNumber := 16#0000_0000,
                    MatchProjectToController := No,
                    CanUseRPIFromProducer := No,
                    InhibitAutomaticFirmwareUpdate := 0,
                    PassThroughConfiguration := EnabledWithAppend,

  DownloadProjectDocumentationAndExtendedProperties := Yes)
        MODULE Local (Parent := "Local",
               ParentModPortId := 1,
               CatalogNumber := "1756-L73",
               Vendor := 1,
               ProductType := 14,
               ProductCode := 94,
               Major := 22,
               Minor := 1,
               PortLabel := "RxBACKPLANE",
               ChassisSize := 7,
               Slot := 0,
               Mode := 2#0000_0000_0000_0001,
               CompatibleModule := 0,
               KeyMask := 2#0000_0000_0001_1111)
        END_MODULE
        TAG
        END_TAG

        PROGRAM MainProgram (MAIN := "MainRoutine",
                             MODE := 0, DisableFlag := 0)
               TAG
               END_TAG

               ROUTINE MainRoutine
               END_ROUTINE

        END_PROGRAM
        TASK MainTask (Type := CONTINUOUS,
      Rate := 10, Priority := 10, Watchdog := 500,
             DisableUpdateOutputs := No, InhibitTask := No)
               MainProgram;
        END_TASK
        PARAMETER_CONNECTION
        END_PARAMETER_CONNECTION

  CONFIG ASCII(XONXOFFEnable := 0,
               DeleteMode := 0, EchoMode := 0,
```

```
                TerminationChars := 65293, AppendChars := 2573,
                BufferSize := 82)
END_CONFIG

CONFIG ControllerDevice END_CONFIG

CONFIG CST(SystemTimeMasterID := 0) END_CONFIG

CONFIG DF1(DuplicateDetection := 1,
    ErrorDetection := BCC Error, EmbeddedResponseEnable := 0,
    DF1Mode := Pt to Pt, ACKTimeout := 50,
    NAKReceiveLimit := 3, ENQTransmitLimit := 3,
    TransmitRetries := 3, StationAddress := 0,
    ReplyMessageWait := 5, PollingMode := 1,
    MasterMessageTransmit := 0, NormalPollNodeFile := "<NA>",
    NormalPollGroupSize := 0, PriorityPollNodeFile := "<NA>",
    ActiveStationFile := "<NA>", SlavePollTimeout := 3000,
    EOTSuppression := 0, MaxStationAddress := 31,
    TokenHoldFactor := 1, EnableStoreFwd := 0,
    StoreFwdFile := "<NA>")
END_CONFIG

CONFIG FileManager END_CONFIG

CONFIG SerialPort(BaudRate := 19200,
           Parity := No Parity, DataBits := 8 Bits of Data,
           StopBits := 1 Stop Bit, ComDriverId := DF1,
           PendingComDriverId := DF1, RTSOffDelay := 0,
           RTSSendDelay := 0, ControlLine := No Handshake,
           PendingControlLine := No Handshake,
           RemoteModeChangeFlag := 0,
           PendingRemoteModeChangeFlag := 0,
           ModeChangeAttentionChar := 27,
           PendingModeChangeAttentionChar := 27,
           SystemModeCharacter := 83,
           PendingSystemModeCharacter := 83,
           UserModeCharacter := 85,
           PendingUserModeCharacter := 85,
           DCDWaitDelay := 0)
END_CONFIG

CONFIG WallClockTime(LocalTimeAdjustment := 0, TimeZone := 0)
CONFIG InternetProtocol [(Internet_Protocol_Attributes)]
END_ CONFIG
CONFIG EthernetPort1 [(Internet_Protocol_Attributes)]
END_ CONFIG
CONFIG EthernetPort2 [(Internet_Protocol_Attributes)]
CONFIG EthernetNetwork [(Ethernet_Network_Attributes)]
END_CONFIG
```

```
END_CONTROLLER
```

**L5K Safety CONTROLLER Example**

```
CONTROLLER example_safety_controller (Description := "Safety
Project",
   ProcessorType := "1756-L73S",
   Major := 22,
   TimeSlice := 20,
   ShareUnusedTimeSlice := 1,
   RedundancyEnabled := 0,
   KeepTestEditsOnSwitchOver := 0,
   DataTablePadPercentage := 50,
   SecurityCode := 0,
   ChangesToDetect := 16#ffff_ffff_ffff_ffff,
   SFCExecutionControl := "CurrentActive",
   SFCRestartPosition := "MostRecent",
   SFCLastScan := "DontScan",
   SerialNumber := 16#0000_0000,
   MatchProjectToController := No,
   CanUseRPIFromProducer := No,
   SafetyLocked := No,
   SignatureRunModeProtect := No,
   ConfigureSafetyIOAlways := No,
   InhibitAutomaticFirmwareUpdate := 0,
   PassThroughConfiguration := EnabledWithAppend,
   DownloadProjectDocumentationAndExtendedProperties := Yes)
       MODULE Local (Parent := "Local",
                     ParentModPortId := 1,
                     CatalogNumber := "1756-L73S",
                     Vendor := 1,
                     ProductType := 14,
                     ProductCode := 148,
                     Major := 22,
                     Minor := 1,
                     PortLabel := "RxBACKPLANE",
                     ChassisSize := 7,
                     Slot := 0,
                     Mode := 2#0000_0000_0000_0001,
                     CompatibleModule := 0,
                     KeyMask := 2#0000_0000_0001_1111,
                     SafetyNetwork := 16#0000_3c77_0315_5105)
       END_MODULE

       MODULE example_safety_controller:Partner
              (Parent := "Local", ParentModPortId := 1,
               CatalogNumber := "1756-L7SP", Vendor := 1,
               ProductType := 14, ProductCode := 146,
               Major := 22, Minor := 1,
```

```
        PortLabel := "RxBACKPLANE", Slot := 1,
        Mode := 2#0000_0000_0000_0000,
        CompatibleModule := 0,
        KeyMask := 2#0000_0000_0001_1111,
        SafetyNetwork := 16#0000_0000_0000_0000)
END_MODULE

TAG
END_TAG

PROGRAM MainProgram (Class := Standard,
        MAIN := "MainRoutine", MODE := 0,
        DisableFlag := 0)
       TAG
       END_TAG

       ROUTINE MainRoutine
       END_ROUTINE

END_PROGRAM

PROGRAM SafetyProgram (Class := Safety,
        MAIN := "MainRoutine", MODE := 0,
        DisableFlag := 0)
       TAG
       END_TAG

       ROUTINE MainRoutine
       END_ROUTINE

END_PROGRAM

TASK MainTask (Type := CONTINUOUS,
        Class := Standard, Rate := 10,
        Priority := 10, Watchdog := 500,
        DisableUpdateOutputs := No, InhibitTask := No)
       MainProgram;
END_TASK

TASK SafetyTask (Type := PERIODIC,
        Class := Safety, Rate := 20,
        Priority := 10, Watchdog := 20,
        DisableUpdateOutputs := No, InhibitTask := No)
       SafetyProgram;
END_TASK

PARAMETER_CONNECTION
END_PARAMETER_CONNECTION
```

```
CONFIG ASCII(XONXOFFEnable := 0,
            DeleteMode := 0, EchoMode := 0,
            TerminationChars := 65293,
            AppendChars := 2573, BufferSize := 82)
END_CONFIG

CONFIG ControllerDevice END_CONFIG

CONFIG CST(SystemTimeMasterID := 0) END_CONFIG

CONFIG DF1(DuplicateDetection := 1,
          ErrorDetection := BCC Error,
          EmbeddedResponseEnable := 0,
          DF1Mode := Pt to Pt, ACKTimeout := 50,
          NAKReceiveLimit := 3, ENQTransmitLimit := 3,
          TransmitRetries := 3, StationAddress := 0,
          ReplyMessageWait := 5, PollingMode := 1,
          MasterMessageTransmit := 0,
          NormalPollNodeFile := "<NA>",
          NormalPollGroupSize := 0,
          PriorityPollNodeFile := "<NA>",
          ActiveStationFile := "<NA>",
          SlavePollTimeout := 3000, EOTSuppression := 0,
          MaxStationAddress := 31, TokenHoldFactor := 1,
          EnableStoreFwd := 0, StoreFwdFile := "<NA>")
END_CONFIG

CONFIG FileManager END_CONFIG

CONFIG SerialPort(BaudRate := 19200,
            Parity := No Parity, DataBits := 8 Bits of Data,
            StopBits := 1 Stop Bit, ComDriverId := DF1,
            PendingComDriverId := DF1, RTSOffDelay := 0,
            RTSSendDelay := 0, ControlLine := No Handshake,
            PendingControlLine := No Handshake,
            RemoteModeChangeFlag := 0,
            PendingRemoteModeChangeFlag := 0,
            ModeChangeAttentionChar := 27,
            PendingModeChangeAttentionChar := 27,
            SystemModeCharacter := 83,
            PendingSystemModeCharacter := 83,
            UserModeCharacter := 85,
            PendingUserModeCharacter := 85, DCDWaitDelay := 0)
END_CONFIG

CONFIG WallClockTime(LocalTimeAdjustment := 0,
                           TimeZone := 0)
CONFIG InternetProtocol [(Internet_Protocol_Attributes)]
END_ CONFIG
```

```
CONFIG EthernetPort1 [(Internet_Protocol_Attributes)]
END_ CONFIG
CONFIG EthernetPort2 [(Internet_Protocol_Attributes)]
CONFIG EthernetNetwork [(Ethernet_Network_Attributes)]
END_CONFIG

END_CONTROLLER
```

# Define a Datatype component

## Introduction

This chapter explains the overall structure of the datatype component.

## Datatype component

Datatype components define the data types used in the logic you export.

## L5X datatype structure

```
<DataTypes>
      <DataType [DataType_Attributes]>
            <Description>
                  <![CDATA[ text ]]>
            </Description>
            <EngineeringUnit>
                  <![CDATA[ engineering_unit_text ]]>
            </EngineeringUnit>
            <Members>
                  [member_list]
            </Members>
      </Datatype
</DataTypes>
```

## L5K datatype structure

```
DATATYPE <DataType_name> ([Description := "text",
      EngineeringUnit := "text",
      DataType_Attributes])
      [member_definitions]
END_DATATYPE
```

## Datatype elements

| L5X Item | L5K Item | Description |
|---|---|---|
| N/A | *DataType_name* | The name of the data type.<br><br>In L5X, use a Name attribute on the <DataType> element. |
| Description | Description | User information about the data type. |
| EngineeringUnit | EngineeringUnit | (optional) User-specified description of the unit of the value, such as feet, gallons, and kilos. |
| Members | *member_definitions* | Defines the members of the data structure. |

## Datatype attributes

| Attribute | Description |
|---|---|
| Name | L5X only. Specify the name of the data type.<br><br>In L5K, the name is an element of the data type statement. |
| Family (L5X)<br>FamilyType (L5K) | Specify StringFamily for a string data type.<br>Specify NoFamily for all other data types. |
| Class | L5X only. Type **User**. |

# Datatype member

There are two kinds of datatype members:

- Bit member—a member in which only a single bit of information is accessed.
- Nonbit member—a member that is defined as another data type, such as SINT, INT, DINT, and COUNTER.

## L5X datatype member structure

```
<Members>
      <Member [Member_Attributes]>
            <Description>
                  <![CDATA[ text ]]>
            </Description>
            <EngineeringUnit>
                  <![CDATA[ engineeringunit_unit_text ]]>
            </EngineeringUnit>
            <State0>
                  <![CDATA[ state0_text ]]>
            </State0>
            <State1>
                  <![CDATA[ state1_text ]]>
            </State1>
      </Member>
</Members>
```

## L5K datatype member structure

```
<TypeName> <MemberName> ([Description := "text",
      EngineeringUnit := "text",
      State0 := "text",
      State1 := "text",
      Member_Attributes]);
```

A bit member uses this syntax:
BIT <*BitName*> <*HostMemberName*> : <*BitPosition*> [(*Attributes)*];

## Datatype member elements

| L5X Item | L5K Item | Description |
|---|---|---|
| N/A | *TypeName* | The name of the data type of the member.<br>In L5X, use a DataType attribute on the <Member> element. |
| N/A | *MemberName* | The name of the member.<br>In L5X, use a Name attribute on the <Member> element. |
| Description | Description | User information about the member. |
| EngineeringUnit | EngineeringUnit | (optional) User-specified description of the unit of the value, such as feet, gallons, and kilos. |
| State0 | State0 | (optional) For Boolean member only. User-specified description of what the Zero state of the Boolean value is. |
| State1 | State1 | (optional) For Boolean member only. User-specified description of what the One state of the Boolean value is. |
| N/A | *BitName* | The name of the bit member.<br>In L5X, use a Name attribute on the <Member> element. |
| N/A | *HostMemberName* | The hidden host member of bit members.<br>In L5X, use a Target attribute on the <Member> element. |
| N/A | *BitPosition* | The bit position in the hidden host member.<br>In L5X, use a BitNumber attribute on the <Member> element. |

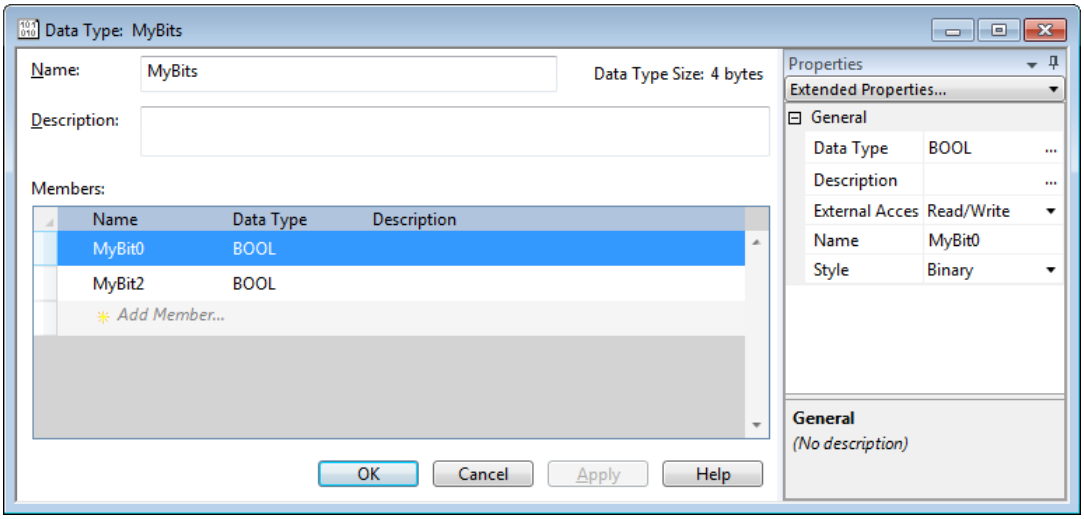
## Datatype member attributes

| Attribute | Description |
|---|---|
| Name | L5X only. Specify the name of the member.<br>In L5K, the member name is an element of the member statement. |
| Dimension | L5X only. Specify the dimensions of the member. Type **0** if atomic or a non-zero value for an array. |
| Radix | Specify **decimal**, **hex**, **octal**, **binary**, **exponential**, **float**, **ASCII**, or **date/time**. |
| Hidden | Specify if the member is a hidden member of the structure. Type **1** or **0** (or **true** or **false** for L5X) |
| Target | The name of the hidden host member. |
| BitNumber | The bit position in the host member. |
| Max | (optional) User-specified maximum value for the member. Only valid for members with non-Boolean atomic datatypes. |
| Min | (optional) User-specified minimum value for the member. Only valid for members with non-Boolean atomic datatypes. |
| External Access | Specify the external access outside of the controller to the member. Specify **Read/Write**, **Read Only**, or **None**. |

## Bit members

All data types are allocated in 8-bit boundaries. A single bit of storage is not allowed, so a member cannot be a BOOL data type. To access a single bit, use the BIT declaration. BIT allows access to a single bit within a host member (a non-bit member).

For example, create a user-defined datatype called MyBits and a tag called MyTag of type MyBits:




| Name | Alias For | Base Tag | Data Type | Description | External Access |
|---|---|---|---|---|---|
| MyTag | | | MyBits | | Read/Write |
| MyTag.MyBit0 | | | BOOL | | Read/Write |
| MyTag.MyBit2 | | | BOOL | | Read/Write |

zzzzzzzzzzMyBits0 is the host member of MyBit0 and MyBit1.

The host member is normally a hidden member because only the bit references are visible when you define a tag of the datatype. Logix Designer appends the z characters to the host member name to prevent a bit overlay error. If you remove the z characters, the datatype can still be imported but it cannot be modified.

Bit members cannot be defined before their host members. Note that BitPosition zero is the least significant bit.

This is the datatype syntax for this example in the L5K format.

```
DATATYPE MyBits (FamilyType := NoFamily)
SINT ZZZZZZZZZZMyBits0 (Hidden := 1);
      BIT MyBit0 ZZZZZZZZZZMyBits0 : 0 (Radix := Binary);
      BIT MyBit1 ZZZZZZZZZZMyBits0 : 1 (Radix := Binary);
END_DATATYPE
```

| | |
|---|---|
| **Important:** | There must be a space between the host member name, colon, and the bit position because type names can contain a colon (for example, I/O structures). The space indicates where the type name ends. |

This is the datatype syntax for this example in the L5X format.

```
- <DataType Name="MyBits" Family="NoFamily" Class="User">
  - <Members>
      <Member Name="ZZZZZZZZZZMyBits0" DataType="SINT" Dimension="0" Radix="Decimal"
        Hidden="true" />
      <Member Name="MyBit0" DataType="BIT" Dimension="0" Radix="Binary" Hidden="false"
        Target="ZZZZZZZZZZMyBits0" BitNumber="0" />
      <Member Name="MyBit1" DataType="BIT" Dimension="0" Radix="Binary" Hidden="false"
        Target="ZZZZZZZZZZMyBits0" BitNumber="1" />
    </Members>
  </DataType>
```

## Datatype guidelines

Observe these guidelines when defining a datatype:

- Datatypes must be defined first within the controller body.
- Datatypes can be defined out of order. For example, if Type1 depends on Type2, Type2 can be defined first.
- Datatypes can be unverified. For example if Type1 depends on Type2 and Type2 is never defined, then Type1 will be accessible as an unverified type. Type2 will be typeless type. Tags of Type1 may be created but not of Type2.
- Datatype members can be arrays but only one dimension is allowed.
- These datatypes cannot be used in a user-defined datatype:
  - ALARM_ANALOG
  - ALARM_DIGITAL
  - AXIS types
  - COORDINATE_SYSTEM
  - MOTION_GROUP
  - MESSAGE
  - MODULE
- If one user-defined datatype references a second user-defined data type defined in the file, the second user-defined datatype appears before the first one in the import/export file.

# Examples

**L5X DataType example**

```
<DataType Name="SampleDT" Family="NoFamily" Class="User" UId="23e0ab2b">
  <Members>
    <Member Name="Sample_DINT_Member" DataType="DINT" Dimension="0"
      Radix="Decimal" Hidden="false">
      <Description>
        <![CDATA[ This is a DINT member of the UDT ]]>
      </Description>
    </Member>
    <Member Name="ZZZZZZZZZZSampleDT1" DataType="SINT" Dimension="0"
      Radix="Decimal" Hidden="true" />
    <Member Name="Sample_BOOL_Member" DataType="BIT" Dimension="0"
      Radix="Decimal" Hidden="false" Target="ZZZZZZZZZZSampleDT1"
      BitNumber="0">
      <Description>
        <![CDATA[ This is a BOOL Member of the UDT ]]>
      </Description>
    </Member>
  </Members>
</DataType>
```

**L5K DATATYPE example**

```
DATATYPE MyStructure (FamilyType := NoFamily)
       DINT x;
       TIMER y[3] (Radix := Decimal);
       SINT MyFlags (Hidden :=1);
       BIT aBit0 MyFlags : 0 (Radix := Binary);
       BIT aBit1 MyFlags : 1 (Radix := Binary);
END_DATATYPE
```

# Define a module component

## Introduction

This chapter explains the overall structure of the module component.

## Module component

A module component defines any modules used by logic you export. For example, the module component can contain I/O modules referenced by I/O tags, modules accessed by GSV/SSV instructions, or controllers referenced in consumed tags.

## L5X module structure

```
<Modules>
      <Module [Module_Attributes]>
            <Description>
                  <![CDATA[ text ]]>
            </Description>
            <EKey [Ekey_Attributes]/>
            <Ports>
                  <Port [Port_Attributes]
                   <Bus [Bus_Attributes]/>
            </Port>
            </Ports>
            <Communications [Communications_Attributes]>
                  <ConfigTag [ConfigTag_Attributes]>
                        [data]
                  </ConfigTag>
                  <ConfigScript [ConfigScript_Attributes]>
                        [data]
                  </ConfigScript>
                  [connections]
            </Communications>
            <ExtendedProperties
[ExtendedProperties_Attributes]>
                        [extendedpropertiesdata]
            </ExtendedProperties>
      </Module>
</Modules>
```

## L5K MODULE structure

```
MODULE <device_name> [(Description := "text",
      Module_Attributes)]
      [ConfigData := <initial_value>;]
      [ConfigScript:=<initial_value>;]
      [ExtendedProp := <text>]
      [connection_list]
END_MODULE
```

## Module elements

| L5X Item | L5K Item | Description |
|---|---|---|
| N/A | *device_name* | The name of the module.<br><br>In L5X, use a Name attribute on the <Module> element. |
| Description | Description | User information about the module. |
| EKey | N/A | Keying information for the module.<br><br>In L5K, this information is in the CompatibleModule and KeyMask attributes on the MODULE. |
| Ports | N/A | Port information for the module, which is the physical connector for the module that attaches the module to the bus. Each module has at least one port. |
| ConfigTag or ConfigData | ConfigData | Operating characteristics of the module.<br><br>In L5X, the data for the tag is defined with <Data> elements. See Chapter 1 Data Formats on page 43 for more information. |
| ExtendedProperties | ExtendedProp | Additional profile data stored in the controller in an XML format. |
| *connections* | *connection_list*<br>(zero or more CONNECTION entries) | Connection characteristics for the module. |

## Module attributes

| Attribute | Description |
|---|---|
| Name | L5X only. Specify the name of the module.<br><br>In L5K the name is an element of the statement. |
| Parent | L5K only. If this module is a child to another module, specify the name of the parent module. The parent module must be defined before any child module. |
| ParentModule | L5X only. If this module is a child to another module, specify the name of the parent module. The parent module must be defined before any child module. |
| ParentModPortID (L5K)<br>ParentModPortID (L5X) | If this module is a child to another module, specify the number of the port on the parent module that connects to this child module. The parent module must be defined before any child module. |
| CatalogNumber | Specify the catalog number of the module. |
| Vendor | Specify the vendor of the module. A number **1** indicates Allen-Bradley®.<br><br>For L5X, this attribute is on the <EKey> element if the State attribute has a value of *Custom*. |

| Attribute | Description |
|---|---|
| ProductType | Specify the product type of the module.<br><br>For L5X, this attribute is on the <EKey> element if the State attribute has a value of *Custom*. |
| ProductCode | Specify the product code of the module.<br><br>For L5X, this attribute is on the <EKey> element if the State attribute has a value of *Custom*. |
| Major | Specify the major revision number (1...127) of the module.<br><br>For L5X, this attribute is on the <EKey> element if the State attribute has a value of *Custom*. |
| Minor | Specify the minor revision number (1...255) of the module.<br><br>For L5X this attribute is on the <EKey> element if the State attribute has a value of *Custom*. |
| UserDefinedVendor | Specify the vendor of a non-Allen-Bradley module. Type a number to indicate the vendor. |
| UserDefinedProductType | Specify the product type of a non-Allen-Bradley module. |
| UserDefinedProductCode | Specify the product code of a non-Allen-Bradley module. |
| UserDefinedMajor | Specify the major revision number (1...127) of a non-Allen-Bradley module. |
| UserDefinedMinor | Specify the minor revision number (1...255) of a non-Allen-Bradley module. |
| PortLabel | L5K only. Specify the port used to reach this module. The port label is either RxBACKPLANE for modules in a chassis or a text string for modules on a network. |
| ChassisSize | L5K only. Specify the number of slots in the chassis.<br><br>For L5X, use the Size attribute on the <Bus> element. |
| Slot | L5K only. Specify the slot number where the module is in the chassis.<br><br>For L5X, use the Address attribute on the <Port> element. |
| NodeAddress | L5K only. Specify the node address (1...99) on the network with the Ethernet IP address or host name).<br><br>For L5X, use the Address attribute on the <Port> element. |
| Group | L5K only. If the module is a remote I/O module, specify the starting group (0...7). For a block-transfer module, this is the module group number under the remote I/O adapter. |
| CommMethod | Specify the method of connecting to the module.<br><br>For L5X, this attribute is on the <Communications> element. |
| ConfigMethod | Specify the method of configuring the module. |
| Mode | Select a specific mode by setting the appropriate bit.<br>**Set:** **For:**<br>0 Do not inhibit the module and a fault in the module does not cause a major fault in the controller<br>1 Fault in the module causes a major fault in the controller<br>4 Inhibit the module<br>5 Both inhibit the module and a fault in the module causes a major fault in the controller |
| CompatibleModule | L5K only. Specify whether to connect to a compatible module based on the minor revision (*value* = 1) or an exact match or disabled keying of the module (*value* = 0).<br>If you specify exact for KeyMask, set CompatibleModule to **0**.<br>If you specify compatible for KeyMask, set CompatibleModule to **1**. |

| Attribute | Description |
|---|---|
| KeyMask | L5K only. Specify whether to connect to the exact module that matches the electronic keying information of vendor, product code, product type, major revision, and minor revision. No keying will connect to any module.<br>**Specify :** **To:**<br>2#0000_0000_0000_0000 Disable keying<br>2#0000_0000_0001_1111 Require a replacement module to be compatible<br>2#0000_0000_0001_1111 Require a replacement module to be an exact match<br>The values for compatible module and for exact match are the same because this attribute is used in conjunction with CompatibleModule to distinguish between compatible module or exact match. |
| State | L5X only. This attribute is on the <EKey> element. Type **CompatibleModule**, **ExactMatch**, **Disabled**, or **Custom**. |
| PrimCxnInputSize | Specify the size of the input data associated with the primary connection (0...500 bytes).<br>For L5X, this attribute is on the <Communications> element. |
| PrimCxnOutputSize | Specify the size of the output data associated with the primary connection (0...496 bytes).<br>For L5X, this attribute is on the <Communications> element. |
| SecCxnInputSize | Specify the size of the input data associated with the secondary connection (0...500 bytes). Typically, there is one I/O connection on a module (primary connection). If there are two, the second connection is the secondary connection.<br>For L5X, this attribute is on the <Communications> element. |
| SecCxnOutputSize | Specify the size of the output data associated with the secondary connection (0...496 bytes). Typically, there is one I/O connection on a module that is the primary connection. If there are two, the second connection is the secondary connection.<br>For L5X, this attribute is on the <Communications> element. |
| ChABaud | L5K only. For a 1756-DHRIO module, specify the baud rate for channel A. Type **57.6**, **115.2**, or **230.4**.<br>For L5X, use the Baud attribute on the <Bus> element. |
| ChBBaud | L5K only. For a 1756-DHRIO module, specify the baud rate for channel B. Type **57.6**, **115.2**, or **230.4**.<br>For L5X, use the Baud attribute on the <BusControlNet> elemEtherNet/IPent. |
| DtlsFileName | Specify the file name associated with a DriveExecutive™ project. DriveExecutive configures drives on ControlNet™ and EtherNet/IP™ networks. |
| ConfigCode | Specify the value that represents the drive rating of the drive. Select this rating on the **Power** tab in a DriveExecutive project for drives on ControlNet and EtherNet/IPnetworks. |
| ControlNetSignature | This value (hexadecimal) is exported only for the purpose of doing a file compare. This value is ignored on import. |
| SafetyNetwork | If the module is in a safety controller system, specify the 6-byte hexadecimal number of the safety network. |
| SafetyEnabled | A flag only in modules that can be configured as safety or standard. Type **true** if the module is a safety module. |
| RSNetWorxFileName | L5K only. Specify the file name of an associated RSNetWorx project file. |
| Inhibited | L5X only. If the module is inhibited, type **true**. If the module is not inhibited, type **false**. |
| MajorFault | L5X only. Specify if the controller generates a major fault if the connection to the module is lost in run mode (**true** or **false**). |
| ConfigSize | L5X only. This attribute is on the <ConfigTag> or <ConfigData> element. Specify the size of the Config Tag or Config Data. |
| Id | L5X only. This attribute is on the <Port> element. It uniquely identifies the port. |
| Address | L5X only. This attribute is on the <Port> element. Specify the node number, slot number, or IP address/host name. |

| Attribute | Description |
|---|---|
| Type | L5X only. This attribute is on the <Port> element. Defines the type of the module |
| Upstream | L5X only. This attribute is on the <Port> element. It is determined by the I/O topology of the module. Specify **true** for Upstream or **false** for downstream. |
| NATActualAddress | The IP address of a safety I/O module or controller specified by the user as the actual address on the network of the module.<br>For L5X, this attribute is on the <Port> element. |
| ConnectorOffset | L5X only. This attribute is on the <Port> element. |
| Width | L5X only. This attribute is on the <Port> element. |
| Size | L5X only. This attribute is on the <Bus> element. For a sizable chassis, specify the chassis size.<br><br>In L5K, this attribute is the ChassisSize attribute on the MODULE. |
| Baud | L5X only. This attribute is on the <Bus> element. Specify the baud rate (57.6, 115.2, or 230.4).<br><br>In L5K, this attribute is the ChABaud or ChBBaud attribute on the MODULE. |
| ShutdownParentOnFault | Indicates the parent device is shut down when this module faults. |
| DrivesADCMode | Sets or clears the Drives ADC mode bit.<br><br>Setting to true on a non-ADC causes the drive to fail. |
| DrivesADCEnabled | Indicates that Automatic Device Configuration is enabled for this device.<br><br>Can be set true only when the DrivesADCMode is set true. |
| UserDefinedCatalogNumber | Used to persist the Catalog Number for drive peripherals.<br><br>You do not have to modify this value. |
| Constant | Specify whether the value is a constant value or a dynamic value. For L5K, specify **yes** for a constant value or **no** for a dynamic value. For L5X, specify **true** or **false**. |
| PermissionSet | Name of the set of permissions, configured in FactoryTalk Security, to apply to this object. |

## Module attributes in a safety controller system

In a safety controller system, the module component for the safety partner follows the module component for the primary safety controller. All of the attributes of the safety partner are determined based on those of the primary safety controller.

The module component for the primary safety controller follows the structure previously described. The safety partner module uses these attributes:

- Parent
- ParentModPortID
- CatalogNumber (1756-LSP)
- Vendor
- ProductType
- ProductCode
- Major
- Minor
- Mode
- PortLabel
- Slot
- CompatibleModule
- Key
- Mask
- SafetyNetwork
- SafetyEnabled
- NATActualAddress

## Module connection

### L5X connection structure

```
<Connections>
      <Connection [Connection_Attributes]>
            <InputTag>
                  data
            </InputTag>
            <OutputTag>
                  data
            </OututTag>
      </Connection>
      <RackConnection [RackConnection_Attributes]>
            <InAliasTag>
                  data
            </InAliasTag>
            <OutAliasTag>
                  data
            </OutAliasTag>
      </RackConnection>
</Connections>
```

### L5K CONNECTION structure

```
CONNECTION <connection_name> [(Connection_Attributes)]
      [InputData := <value_list>;]
      [InputForceData := <value_list>;]
      [OutputData := <value_list>;]
      [OutputForceData := <value_list>;]
END_CONNECTION
```

## Connection elements

| L5X Item | L5K Item | Description |
|---|---|---|
| N/A | connection_name | The name of the connection.<br><br>In L5X, use a Name attribute on the <Connection> element. |
| InputTag or InputData | InputData | Input channel data.<br><br>In L5X, the data for the input tag is defined with <Data> elements. See Chapter 1 Data Formats on page 43 for more information. |
| InputTag or InputData | InputForceData | Forcing information for the input channel.<br><br>In L5X, the force data for the input tag is defined with <ForceData> elements. See Chapter 1 Data Formats on page 43 for more information. |
| OutputTag or OutputData | OutputData | Output channel data.<br><br>In L5X, the data for the output tag is defined with <Data> elements. See Chapter 1, Data Formats on page 43 for more information. |
| OutputTag or OutputData | OutputForceData | Forcing information for the output channel.<br><br>In L5X, the force data for the output tag is defined with <ForceData> elements. See Chapter 1 Data Formats on page 43 for more information. |

For details on the data in the connection list, see the user manual for the I/O module. The connection list data depends on the I/O module and the configuration for that module.

In L5K format, forces appear as arrays of bytes under the InputForceData and OutputForceData attributes of the connection list. In L5X format, forces appear as arrays of bytes under the <ForceData> elements in the <InputTag> and <OutputTag> elements.

| | |
|---|---|
| **Important:** | Do not modify forces in the import/export file. Use the programming software to enter and enable forces. |

## Module connection attributes

| Attribute | Description |
|---|---|
| Name | L5X only. Specify the name of the connection.<br><br>In L5K, the name is an element of the statement. |
| Rate | L5K only. Specify the requested packet interval (RPI) rate in microseconds. |
| RPI | L5X only. Specify the requested packet interval rate in microseconds. |
| InputCxnPoint | Specify the input connection point for the primary connection (0...255). |
| InputSize | Specify the input size (0...255). |
| OutputCxnPoint | Specify the output connection point for the primary connection (0...255). |

| Attribute | Description |
|---|---|
| OutputSize | Specify the output size (0...255). |
| Unicast | Specify if the EtherNet/IP connection is unicast. For L5K, specify **yes** for unicast or **no** to remain multicast. For L5X, specify **true** or **false**.<br>On export, only appears if the path to the unicast supported I/O module crosses an EtherNet/IP network. |
| EventID | Specify the event ID if used in conjunction with an event task. |
| ControlNetScheduled | Specify how the connection is scheduled. Specify **yes** to schedule over ControlNet or **no** to connect unscheduled. This attribute is only used if the path from the module to the controller uses ControlNet. For L5X format specify **true** or **false**. |
| Type | L5X only. Specify the type of connection: **Input**, **Output**, **MotionSync**, **MotionAsync**, **MotionEvent**, **SafetyInput**, or **SafetyOutput**. |
| Priority | Indicates the rank of the input production.<br><br>Valid values = **Scheduled** (default) or **High**. |
| InputConnectionType | Indicates the type of input production.<br><br>Valid values = **Multicast** (default) or **Unicast**. |
| OutputRedundantOwner | Indicates if the output production is a redundant owner. |
| InputProductionTrigger | Indicates the input production trigger.<br><br>Valid values = **Cyclic**, **COS**, or **Application**. |
| ConnectionPath | Indicates the target connection path. |
| InputTagSuffix | Identifies the suffix for the Input Tag. |
| OutputTagSuffix | Identifies the suffix for the Output Tag |
| Constant | Specify whether the value for an Input or an Output parameter is a constant value or it can change. For L5K, specify **yes** for a constant value or **no** for a dynamic value. For L5X, specify **true** or **false**. |
| PermissionSet | Name of the set of permissions, configured in FactoryTalk Security, to apply to this Input/Output Tag. |

## Module connection attributes in a safety controller system

A module connection in a safety controller system has these attributes, in addition to the module connection attributes previously described.

| Attribute | Description |
|---|---|
| TimeoutMultiplier | Specify the timeout multiplier (default = 2) for a safety controller system. This value determines the RPIs of time to wait for a packet before declaring a time out. This translates into the number of messages that may be lost before declaring a connection error. A Timeout Multiplier of 1 indicates that no messages may be lost; that is, there must be a packet every RPI. A Timeout Multiplier of 2 indicates that 1 message may be lost; that is, as long as a packet is seen in 2 times the RPI, no time-out will occur. Type a number from 1...4. |
| NetworkDelayMultiplier | Specify the network delay multiplier (default = 100%) for a safety controller control system. This value lets you reduce or increase the connection reaction time limit in cases where the transport time of the message is significantly less or more than the RPI. This may be the case when the RPI of an output connection is the same as a lengthy task period. Type a percentage from 10...600. |
| ReactionTimeLimit | Specify the connection reaction time limit (0...5500032) for a safety controller system. The Logix Designer application calculates the connection reaction time limit as a function of the RPI, timeout multiplier, and network delay multiplier. The connection reaction time limit is automatically recalculated if any of the values change. |

| Attribute | Description |
|---|---|
| MaxObservedNetworkDelay | L5X only. The MaxObservedNetworkDelay is a measure of the longest time data for a safety connection is delayed from transporting the safety packets over the network. This attribute is exported for informational purposes only and is ignored on import. |

## Module guidelines

Observe these guidelines when defining a module.

- Attributes can be in any order. They export in the order defined.
- A parent module must be defined before any definitions of its child modules.

# Examples

**L5X module example**

```
- <Modules>
  - <Module Name="Local" CatalogNumber="1756-L62" Vendor="1" ProductType="14"
      ProductCode="55" Major="17" Minor="1" ParentModule="Local" ParentModPortId="1"
      Inhibited="false" MajorFault="true">
      <EKey State="ExactMatch" />
    - <Ports>
      - <Port Id="1" Address="0" Type="ICP" Upstream="false">
          <Bus Size="10" />
        </Port>
      </Ports>
    </Module>
  - <Module Name="DHRIO_Module" CatalogNumber="1756-DHRIO/B" Vendor="1"
      ProductType="12" ProductCode="18" Major="2" Minor="1" ParentModule="Local"
      ParentModPortId="1" Inhibited="false" MajorFault="false">
      <EKey State="CompatibleModule" />
    - <Ports>
        <Port Id="1" Address="1" Type="ICP" Upstream="true" />
      - <Port Id="2" Type="RIO" Upstream="false">
          <Bus Baud="57.6" />
        </Port>
      - <Port Id="3" Type="RIO" Upstream="false">
          <Bus Baud="57.6" />
        </Port>
      </Ports>
    - <Communications CommMethod="536870913">
      - <Connections>
        - <Connection Name="Standard" RPI="25000" Type="Input" EventID="0"
            ProgrammaticallySendEventTrigger="false">
          - <InputTag>
              <ForceData>00 00 00 00 00 00 00 00 00 00 00 00</ForceData>
            - <Data Format="Decorated">
              - <Structure DataType="AB:1756_DHRIO:I:0">
                  <DataValueMember Name="CHA_Status" DataType="SINT"
                    Radix="Decimal" Value="0" />
```

```
                    <DataValueMember Name="CHB_Status" DataType="SINT"
                      Radix="Decimal" Value="0" />
                  </Structure>
                </Data>
              </InputTag>
            </Connection>
          </Connections>
        </Communications>
      </Module>
    - <Module Name="Output_Module" CatalogNumber="1756-OB16D" Vendor="1"
        ProductType="7" ProductCode="4" Major="3" Minor="1" ParentModule="Local"
        ParentModPortId="1" Inhibited="false" MajorFault="false">
        <EKey State="CompatibleModule" />
      - <Ports>
          <Port Id="1" Address="2" Type="ICP" Upstream="true" />
        </Ports>
      - <Communications CommMethod="536870914">
        - <ConfigTag ConfigSize="40">
            <Data>2C 00 00 00 13 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
              00 00 00 00 00 00 00 00 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 00 00 00
              00</Data>
          - <Data Format="Decorated">
            - <Structure DataType="AB:1756_DO_DC_Diag:C:0">
                <DataValueMember Name="ProgToFaultEn" DataType="BOOL"
                  Value="0" />
                <DataValueMember Name="FaultMode" DataType="DINT" Radix="Binary"
                  Value="2#0000_0000_0000_0000_0000_0000_0000_0000" />
                <DataValueMember Name="FaultValue" DataType="DINT" Radix="Binary"
```

```
                Value="2#0000_0000_0000_0000_0000_0000_0000_0000" />
              <DataValueMember Name="ProgMode" DataType="DINT" Radix="Binary"
                Value="2#0000_0000_0000_0000_0000_0000_0000_0000" />
              <DataValueMember Name="ProgValue" DataType="DINT" Radix="Binary"
                Value="2#0000_0000_0000_0000_0000_0000_0000_0000" />
              <DataValueMember Name="FaultLatchEn" DataType="DINT"
                Radix="Binary"
                Value="2#0000_0000_0000_0000_1111_1111_1111_1111" />
              <DataValueMember Name="NoLoadEn" DataType="DINT" Radix="Binary"
                Value="2#0000_0000_0000_0000_1111_1111_1111_1111" />
              <DataValueMember Name="OutputVerifyEn" DataType="DINT"
                Radix="Binary"
                Value="2#0000_0000_0000_0000_1111_1111_1111_1111" />
            </Structure>
          </Data>
        </ConfigTag>
      - <Connections>
        - <Connection Name="Diagnostic" RPI="20000" Type="Output" EventID="0"
            ProgrammaticallySendEventTrigger="false">
          - <InputTag>
              <ForceData>00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                00 00 00 00 00 00 00 00 00 00 00</ForceData>
            - <Data Format="Decorated">
              - <Structure DataType="AB:1756_DO_DC_Diag:I:0">
                  <DataValueMember Name="Fault" DataType="DINT"
                    Radix="Binary"
                    Value="2#0000_0000_0000_0000_0000_0000_0000_0000" />
                  <DataValueMember Name="Data" DataType="DINT" Radix="Binary"
                    Value="2#0000_0000_0000_0000_0000_0000_0000_0000" />
                - <ArrayMember Name="CSTTimestamp" DataType="DINT"
                    Dimensions="2" Radix="Decimal">
                    <Element Index="[0]" Value="0" />
                    <Element Index="[1]" Value="0" />
                  </ArrayMember>
                  <DataValueMember Name="FuseBlown" DataType="DINT"

                    Radix="Binary"
                    Value="2#0000_0000_0000_0000_0000_0000_0000_0000" />
                  <DataValueMember Name="NoLoad" DataType="DINT"
                    Radix="Binary"
                    Value="2#0000_0000_0000_0000_0000_0000_0000_0000" />
                  <DataValueMember Name="OutputVerifyFault" DataType="DINT"
                    Radix="Binary"
                    Value="2#0000_0000_0000_0000_0000_0000_0000_0000" />
                </Structure>
              </Data>
            </InputTag>
          - <OutputTag>
              <Data>00 00 00 00</Data>
              <ForceData>00 00 00 00 00 00 00 00 00 00 00 00</ForceData>
            - <Data Format="Decorated">
              - <Structure DataType="AB:1756_DO:O:0">
                  <DataValueMember Name="Data" DataType="DINT" Radix="Binary"
                    Value="2#0000_0000_0000_0000_0000_0000_0000_0000" />
                </Structure>
              </Data>
            </OutputTag>
          </Connection>
        </Connections>
      </Communications>
    </Module>
  </Modules>
```

**L5X Safety Partner Module Example**

```
- <Module Name="SafetyProject:Partner" CatalogNumber="1756-LSP" Vendor="1"
    ProductType="14" ProductCode="69" Major="17" Minor="1" ParentModule="Local"
    ParentModPortId="1" Inhibited="false" MajorFault="false"
    SafetyNetwork="16#0000_0000_0000_0000">
  - <Description>
      <![CDATA[ My Safety Project!  ]]>
    </Description>
    <EKey State="ExactMatch" />
  - <Ports>
      <Port Id="1" Address="4" Type="ICP" Upstream="true" Width="0" />
    </Ports>
  </Module>
```

**L5K MODULE example**

```
MODULE Local (Parent := "Local",
              ParentModPortId := 1,
              CatalogNumber := "1756-L62",
              Vendor := 1,
              ProductType := 14,
              ProductCode := 55,
              Major := 17,
              Minor := 1,
              PortLabel := "RxBACKPLANE",
              ChassisSize := 10,
              Slot := 0,
              Mode := 2#0000_0000_0000_0001,
              CompatibleModule := 0,
              KeyMask := 2#0000_0000_0001_1111)
END_MODULE

MODULE DHRIO_Module (Parent := "Local",
                     ParentModPortId := 1,
                     CatalogNumber := "1756-DHRIO/B",
                     Vendor := 1,
                     ProductType := 12,
                     ProductCode := 18,
                     Major := 2,
                     Minor := 1,
                     PortLabel := "RxBACKPLANE",
                     Slot := 1,
                     CommMethod := 536870913,
                     ConfigMethod := 8388609,
                     Mode := 2#0000_0000_0000_0000,
                     CompatibleModule := 1,
                     KeyMask := 2#0000_0000_0001_1111,
                     ChABaud := 57.6,
                     ChBBaud := 57.6)
                  CONNECTION Standard (Rate := 25000,
                                       EventID := 0)
                                    InputData  := [0,0];
                                    InputForceData := [0,0,0,0,0,0,0,0,0,0,0,0,0];
                  END_CONNECTION

END_MODULE

MODULE Output_Module (Parent := "Local",
                      ParentModPortId := 1,
                      CatalogNumber := "1756-OB16D",
                      Vendor := 1,
                      ProductType := 7,
                      ProductCode := 4,
                      Major := 3,
                      Minor := 1,
                      PortLabel := "RxBACKPLANE",
                      Slot := 2,
                      CommMethod := 536870914,
                      ConfigMethod := 8388612,
                      Mode := 2#0000_0000_0000_0000,
                      CompatibleModule := 1,
                      KeyMask := 2#0000_0000_0001_1111)
                   ConfigData  := [44,19,1,0,0,0,0,0,0,0,65535,65535,65535,0];
                   CONNECTION Diagnostic (Rate := 20000,
                                          EventID := 0)
                                    InputData  := [0,0,[0,0],0,0,0,0];
                                    InputForceData :=
[0,0,0,0,0,0,0,0,0,0,0,0,
```

**L5K Safety Partner MODULE example**

```
MODULE SafetyProject:Partner (Description := "My Safety Project!",
                              Parent := "Local",
                              ParentModPortId := 1,
                              CatalogNumber := "1756-LSP",
                              Vendor := 1,
                              ProductType := 14,
                              ProductCode := 69,
                              Major := 17,
                              Minor := 1,
                              PortLabel := "RxBACKPLANE",
                              Slot := 4,
                              Mode := 2#0000_0000_0000_0000,
                              CompatibleModule := 0,
                              KeyMask := 2#0000_0000_0001_1111,
                              SafetyNetwork := 16#0000_0000_0000_0000)
END_MODULE
```

# Define an Add-On Instruction component

## Introduction

This chapter explains the overall structure of the Add-On Instruction component.

## Add-On Instruction component

An Add-On Instruction component defines an Add-On Instruction.

| | |
|---|---|
| **Important:** | The EditedDate attribute of an Add-On Instruction must be updated if the Add-On Instruction is modified by editing an.L5K or.L5X file.<br><br>When you change an Add-On Instruction by manually editing an.L5X file using a text editor and you do not change the EditedDate attribute, the application does not overwrite the Add-On Instruction when you import the file. |

## L5X AddOnInstructionDefinition Structure

```
<AddOnInstructionDefinitions>
      <AddOnInstructionDefinition
[AddOnInstructionDefinition_Attributes]>
            <Description>
            <![CDATA[ text ]]>
            </Description>
            <AdditionalHelpText>
            <![CDATA[ text ]]>
            </AdditionalHelpText>
            <SignatureHistory>
            [history]
            </SignatureHistory>
            <Parameters>
            [parameter]
            </Parameters>
            <LocalTags>
            [local_tag]
            </LocalTags>
            <Routines>
            [routine]
            </Routine>
            <ScanModeRoutine>
            [routine]
            </ScanModeRoutine>
      </AddOnInstructionDefintion>
</AddOnInstructionDefinitions>
```

## L5K ADD_ON_INSTRUCTION_DEFINITION structure

```
ADD_ON_INSTRUCTION_DEFINITION <name>
   [(Description := "text", Attributes)]
   [<HISTORY_ENTRY declaration>]
   [<PARAMETERS declaration>]
   [<LOCAL_TAGS declaration>]
   [<add_on_instruction_routines]
   [<scan_mode_routine>]
END_ADD_ON_INSTRUCTION_DEFINITION
```

## Add-On Instruction elements

| L5X Item | L5K Item | Description |
|---|---|---|
| N/A | name | In L5K, the name of the Add-On Instruction.<br><br>In L5X, use a Name attribute on the <AddOnInstructionDefinition> element. |
| Description | Description | User information about the Add-On Instruction (128 characters maximum). |
| SignatureHistory | HISTORY_ENTRY | Optional element for a sealed Add-On Instruction. |
| AdditionalHelpText | N/A | In L5K, use the AdditonalHelpText attribute.<br><br>For L5X, specify help text for the user help on the Add-On Instruction. |
| Parameters | PARAMETERS | Parameters of the Add-On Instruction. |
| LocalTags | LOCAL_TAGS | Local tags of the Add-On Instruction. |
| Routines | add_on_instruction_ routines | Logic that comprises the Add-On Instruction. Logic can be relay ladder, function block, or structured text. |
| ScanModeRoutine | SCAN_MODE_ROUTINE | Optional element that has the logic for a Prescan routine, Postscan routine, or EnableInFalse routine. |
| EncryptionInfo | ENCRYPTION_INFO | Details of the license based source protection for the lockable object. Only exists for protected Add-On Instructions exported in plain text. |
| EncryptedAOIContent | N/A | Source-protected and locked Add-On Instruction content. Only exists for locked Add-On Instructions exported in plain text. |

## Add-On Instruction attributes

| Attribute | Description |
|---|---|
| Name | L5X only. Specify the name of the Add-On Instruction component.<br>In L5K, the name is an element of the statement. |
| Class | Specify the class of the Add-On Instruction. This attribute applies only to safety projects. Type **Standard** or **Safety**. |
| Revision | Specify the revision of the Add-On Instruction, in the form of MajorRevision.MinorRevision. Each revision number can be 1...65,535. If there is no period, the number is treated as a major revision only. |
| RevisionExtension | Provide additional information about the revision (40 characters maximum). |
| RevisionNote | Provide information about the revision (128 characters maximum). |
| Vendor | Specify the name of the vendor (40 characters maximum) of the Add-On Instruction. |

| Attribute | Description |
|---|---|
| ExecutePrescan | Specify whether to execute the Prescan routine after the Logic is prescanned. Type **1** for yes; type **0** for no. The default is 1 if a Prescan routine exists. |
| ExecutePostscan | Specify whether to execute the Postscan routine after the Logic is postscanned. Type **1** for yes; type **0** for no. The default is 1 if a Postscan routine exists. |
| ExecuteEnableInFalse | Specify whether to execute the EnableInFalse routine when enable is false. Type **1** for yes; type **0** for no. The default is 1 if an EnableInFalse routine exists. |
| CreatedDate | Specify the date the Add-On Instruction was created. |
| CreatedBy | Specify the developer that created the Add-On Instruction. |
| EditedDate | Specify the date the Add-On Instruction was last edited. |
| EditedBy | Specify the developer that edited the Add-On Instruction. |
| SoftwareRevision | Specify the revision of the application last used to edit the Add-On Instruction. The default is the currently open version of the application. |
| AdditionalHelpText | Specify help text specific to the Add-On Instruction. |
| PermissionSet | Name of the set of permissions, configured in FactoryTalk Security, to apply to this object. |
| IsEncrypted | Indicates whether the Add-On Instruction is protected with license-based Source Protection and locked. |
| TrackingGroups | The group of tracked objects to which this item belongs. Components can be marked for tracking to determine whether they have been changed. Version 30 of the Logix Designer application supports only one tracking group. |

## Routines in Add-On Instructions

Enter routines in an Add-On Instruction the same as logic routines. The logic in a single routine must all be in the same programming language, but each routine can be in a different programming language. Program the routines in ladder logic (ROUTINE), function block (FBD_ROUTINE), or structured text (ST_ROUTINE) languages. The Add-On Instruction has predefined routine names that you must use and cannot change.

| Routine Name | Description |
|---|---|
| Logic | Defines the logic for the Add-On Instruction.<br>At the minimum, every Add-On Instruction must have a Logic routine. |
| Prescan | Defines logic to execute during prescan. |
| Postscan | Defines logic to execute during postscan. |
| EnableInFalse | Defines logic to execute when EnableIn is false. |

For example, this structure for an Add-On Instruction uses all four routines.

```
ADD_ON_INSTRUCTION_DEFINITION Example (attributes)
        PARAMETERS
                add_on_instruction_parameters
        END_PARAMETERS

        LOCAL_TAGS
                add_on_instruction_local_tags
        END_LOCAL_TAGS

        FBD_ROUTINE Logic (attributes)
```

```
            function_block_routine_logic
        END_FBD_ROUTINE

        ST_ROUTINE Prescan (attributes)
            structured_text_routine_logic
        END_ST_ROUTINE

        ROUTINE Postscan (attributes)
            ladder_logic_routine_logic
        END_ROUTINE

        FBD_ROUTINE EnableInFalse (attributes)
            function_block_routine_logic
        END_FBD_ROUTINE
END_ADD_ON_INSTRUCTION_DEFINITION
```

If a tag in an Add-On Instruction references a second Add-On Instruction whose definition is also present in the file, the referenced Add-On Instruction definition must appear before the first one in the import/export file.

## Parameters

The parameter component defines Input, Output, and InOut type parameters in the Add-On Instruction. For L5X format, the default data for the local tag is defined with a <DefaultData>. The <DefaultData> element is defined the same as a <Data> element. See Chapter 1 Data Formats on page 43 for more information on the <Data> element format.

The system defined EnableIn input parameter and EnableOut output parameter are defined in the export format so the description may be modified in a.L5K or.L5X format file. The rest of the attributes for the EnableIn and EnableOut parameters that are system defined will be ignored by import even though they are present in the import file.

### L5X parameters structure

```
<Parameters>
    <Parameter [Parameter_Attributes]>
        <Description>
        <![CDATA[ text ]] >
        </Description>
        <Comments>
        <Comment Operand="specifier">
        <![CDATA[ comment_text ]]>
        </Comment>
        </Comments>
        <EngineeringUnits>
        <EngineeringUnit Operand="specifier">
        <![CDATA[ engineering_unit_text ]]>
        </EngineeringUnit>
        </EngineeringUnits>
        <Mins>
        <Min Operand="specifier"> min_value </Min>
```

```
                    </Mins>
                    <Maxes>
                    <Max Operand="specifier"> max_value </Max>
                    </Maxes>
                    <State0s>
                    <State0 Operand="specifier">
                    <![CDATA[ state0_text ]]>
                    </State0>
                    </State0s>
                    <State1s>
                    <State1 Operand="specifier">
                    <![CDATA[ state1_text ]]>
                    </State1>
                    </State1s>
                    <DefaultData [DefaultData_Attributes]>
                    data
                    </DefaultData>
          </Parameter>
</Parameters>
```

## L5K parameters structure

```
PARAMETERS
      <name> : <datatype[array_specification]>[(Description :=
"text",
                Comment := "text",
                EngineeringUnit := "text",
                Max := value,
                Min := value,
                State0 := "text",
                State1 := "text",
                Parameter_Attributes)];
END_PARAMETERS
```

## Parameter elements

| L5X Item | L5K Item | Description |
|---|---|---|
| N/A | *name* | The name of the parameter.<br><br>In L5X, use a Name attribute on the <Parameter> element. |
| N/A | *datatype* | Data type of the parameter.<br>InOut parameters can be atomic (SINT, INT, DINT, and REAL) and compound (user-defined and array) data types. Input and Output parameters can be only atomic (SINT, INT, DINT, and REAL) data types.<br><br>In L5X, use a DataType attribute on the <Parameter> element. |
| N/A | *array_specification* | Dimensional boundaries for an InOut parameter array.<br><br>In L5X, use a Dimensions attribute on the <Parameter> element. |
| Description | Description | User information about the parameter with a 128-character maximum. |
| DefaultData | N/A | The default data values for Input parameters or Output parameters.<br><br>In L5K, use a DefaultData attribute. |

| L5X Item | L5K Item | Description |
|---|---|---|
| Comment | Comment | (optional) User information about specified sub-regions of the parameter.<br><br>Can specify Comment<*specifier*><br>Where the specifier is:<br>*.bitnumber* - for a bit in the parameter<br>*[element]* - for an array element of the parameter<br>*.membername* - for a structure member of the parameter<br><br>There can be multiple comment elements. |
| EngineeringUnit | EngineeringUnit | (optional) User specified description of what the unit of the value is (that is, feet, gallons, kilos).<br><br>Can specify EngineeringUnit <*specifier*><br>Where the specifier is:<br>*.bitnumber* - for a bit in the parameter<br>*[element]* - for an array element of the parameter<br>*.membername* - for a structure member of the parameter<br><br>There can be multiple engineering unit elements. |
| Max | Max | (optional) User specified maximum value about qualified sub-regions of the parameter. Only valid for a parameter's sub-regions, which is a non-Boolean atomic datatypes.<br><br>Can specify Max <*specifier*><br>Where the specifier is:<br>*.bitnumber* - for a bit in the parameter<br>*[element]* - for an array element of the parameter<br>*.membername* - for a structure member of the parameter<br><br>There can be multiple max elements. |
| Min | Min | (optional) User specified minimum value about qualified sub-regions of the parameter. Only valid for a parameter's sub-regions, which is a non-Boolean atomic datatypes.<br><br>Can specify Min <*specifier*><br>Where the specifier is:<br>*.bitnumber* - for a bit in the parameter<br>*[element]* - for an array element of the parameter<br>*.membername* - for a structure member of the parameter<br><br>There can be multiple min elements. |
| State0 | State0 | (optional) for Boolean parameters or sub-regions only. User specified description of what the Zero state of the Boolean value is.<br><br>Can specify State0 <*specifier*><br>Where the specifier is:<br>*.bitnumber* - for a bit in the parameter<br>*[element]* - for an array element of the parameter<br>*.membername* - for a structure member of the parameter<br><br>There can be multiple state0 elements. |

| L5X Item | L5K Item | Description |
|---|---|---|
| State1 | State1 | (optional) for Boolean parameters or sub-regions only. User specified description of what the One state of the Boolean value is.<br><br>Can specify State1 <*specifier*><br>Where the specifier is:<br>*.bitnumber* - for a bit in the parameter<br>*[element]* - for an array element of the parameter<br>*.membername* - for a structure member of the parameter<br><br>There can be multiple state1 elements. |

## Parameters attributes

| Attribute | Description |
|---|---|
| Name | L5X only. Specify the name of the parameter.<br><br>In L5K, the name is an element of the statement. |
| DataType | L5X only. Specify the datatype of the parameter.<br>InOut parameters can be atomic (SINT, INT, DINT, and REAL), compound (user-defined and array) data types object backed (MESSAGE, ALARM, etc.). Input and Output parameters can be only atomic (SINT, INT, DINT, and REAL) data types.<br><br>In L5K, the datatype is an element of the statement. |
| TagType | L5X only. Specify **Base** or **Alias**. |
| AliasFor | L5X only. Name of the base tag that the alias parameter references.<br>Specify LocaTag<specifier><br>Where the *specifier* is a bit (*.bitnumber*), array element ([*element*]), or structure member (*.membername*) of the tag, or any combination such as *[7].Input.0*. |
| Dimensions | L5X only. Specify the dimensions of the datatype.<br><br>In L5K, the dimensions are an element of the statement. |
| Usage | Specify the type of parameter. Type **Input**, **Output**, or **InOut**. |
| Radix | Specify **decimal**, **hex**, **octal**, **binary**, **exponential**, **float**, **ASCII**. |
| Required | Specify whether the parameter is required. Type **1** if the parameter is required; type **0** if the parameter is optional. |
| Constant | Specify whether the value for an Input or an Output parameter is a constant value or it can change. For L5K, specify **yes** for a constant value or **no** for a dynamic value. For L5X, specify **true** or **false**. |
| ExternalAccess | Specify the external access, outside of the controller, to the parameter. Specify **Read/Write**, **Read Only**, or **None**. |
| Max | (optional) User specified maximum value for the parameter. Only valid for parameter with non-Boolean atomic datatype. |
| Min | (optional) User specified minimum value for the parameter. Only valid for parameter with non-Boolean atomic datatype. |
| Visible | Specify whether the parameter is visible on the display for the instructions. Type **1** if the parameter is visible; type **0** if the parameter is not visible. |

| Attribute | Description |
|---|---|
| DefaultData | L5K only. Specify a default value for the parameter. This attribute is not available if you specify Usage as InOut.<br><br>In L5X, the parameter default data is an element of the L5X structure. |

## Signature history

The Signature History stores history entries for an Add-On Instruction. There can be 0...6 entries that are exported in the order they are created. The order in the file is used during import to store them. If you edit the file manually, that order is maintained.

- L5K file—The Signature History is stored in the HISTORY_ENTRY structure.
- L5X file—The Signature History is stored in the <SignatureHistory> structure.

When an Add-On Instruction is sealed, the Signature History is protected and hidden in the ENCODED_DATA section. See Encoded/UnencodedAdd-On Instructions on page 96.

- L5K file—The ENCODED_DATA section is a separate section.
- L5X file—The <EncodedData> section is an element of <AddOnInstructionDefinitions>.

### L5X SignatureHistory structure

```
<SignatureHistory>
      <HistoryEntry [HistoryEntry_Attributes]>
            <Description>
            <![CDATA[ text ]] >
            </Description>
      </HistoryEntry>
</SignatureHistory>
```

### L5K HISTORY_ENTRY structure

```
HISTORY_ENTRY [History_Entry_Attributes)]
END_HISTORY_ENTRY
```

## History entry attributes

| Attribute | Description |
|---|---|
| User | Specifies the identity of the user that created the entry. |
| Timestamp | Specifies the timestamp when the entry was created. The value is a UTC date time, such as 2009-04-01T12:08:00.000Z. |
| SignatureID | Specifies the signature ID for the Add-On Instruction when the entry was created. The value is an 8-digit uppercase hex number, such as 8F44EBA3. |
| Description | User information about the parameter (128 characters maximum). |

## Local tags

The local tags component defines local tags in the Add-On Instruction. The L5K format for defining a local tag is the same format for defining a tag in a program or at controller scope For more details on defining a tag, see Chapter 6 Defining a Tag Component on page 107. For L5X format, you specify the default data for the local tag with a <DefaultData> element. Define the <DefaultData> element the same as a <Data> element. See Chapter 1 Data Formats on page 43 for more information on the <Data> element format.

## L5X LocalTags structure

```
<LocalTags>
      <LocalTag [LocalTag_Attributes]>
            <Description>
            <![CDATA[ text ]] >
            </Description>
            <Comments>
            <Comment Operand="specifier">
            <![CDATA[ comment_text ]]>
            </Comment>
            </Comments>
            <EngineeringUnits>
            <EngineeringUnit Operand="specifier">
            <![CDATA[ engineering_unit_text ]]>
            </EngineeringUnit>
            </EngineeringUnits>
            <Mins>
            <Min Operand="specifier"> min_value </Min>
            </Mins>
            <Maxes>
            <Max Operand="specifier"> max_value </Max>
            </Maxes>
            <State0s>
            <State0 Operand="specifier">
            <![CDATA[ state0_text ]]>
            </State0>
            </State0s>
            <State1s>
            <State1 Operand="specifier">
            <![CDATA[ state1_text ]]>
            </State1>
            </State1s>
            <DefaultData [DefaultData_Attributes]>
            data
            </DefaultData>
      </LocalTag>
</LocalTags>
```

## L5K LOCAL_TAGS structure

```
LOCAL_TAGS
        tag_declaration
END_LOCAL_TAGS
```

## Local tag attributes

Specify these attributes for local tags in L5X format. See Chapter 6 Defining a Tag Component on page 107 for attributes for local tags in L5K format.

| Attribute | Description |
|---|---|
| Name | L5X only. Specify the name of the local tag.<br><br>In L5K, the name is an element of the statement. |
| DataType | L5X only. Specify the datatype of the local tag.<br>Local tags can be atomic (SINT, INT, DINT, and REAL) and compound (for example, user-defined, add-on instruction defined, array) data types. Local tags cannot be object backed data types (for example, MESSAGE, ALARM).<br><br>In L5K, the datatype is an element of the statement. |
| Dimensions | L5X only. Specify the dimensions of the datatype.<br><br>In L5K, the dimensions are an element of the statement. |
| Radix | Specify **decimal**, **hex**, **octal**, **binary**, **exponential**, **float**, **ASCII**. |
| ExternalAccess | Specify the external access, outside of the controller, to the local tag. Specify **Read/Write**, **Read Only**, or **None**. |
| Max | (optional) User specified maximum value for the local tag. Only valid for local tag with non-Boolean atomic datatype. |
| Min | (optional) User specified minimum value for the local tag. Only valid for local tag with non-Boolean atomic datatype. |
| DefaultData | L5K only. Specify a default value for the local tag.<br><br>In L5X, the parameter default data is an element of the L5X structure. |

# Encoded/Unencoded Add-On Instructions

These examples are for protected (encoded) and unprotected (clear text) codes for Add-On Instructions.

If the project contains high-integrity Add-On Instructions, those Add-On Instructions always appear as encoded data components when you export the project.

See Exporting Source-protected Logic on page 28 on page 34 for procedures.

## L5X EncodedData Structure

```
<EncodedData EncodedType= "type", Name="name",
Type="routinetype"
            [,other_attributes]>
      <Description>
            <![CDATA[ text ]] >
      </Description>
      encoded_data
</EncodedData>
```

## L5K ENCODED_DATA Structure

```
ENCODED_DATA [( EncodedType: type, Name:= name,
                Type:= routinetype,
                other_attributes)]
      encoded_data
END_ENCODED_DATA
```

## Encoded data attributes

| L5X Item | L5K Item | Description |
|---|---|---|
| *type* | *type* | The type of data encoded.<br><br>In L5K, specify **ADD_ON_INSTRUCTION_DEFINITION**<br>In L5X, specify **AddOnInstructionDefinition** |
| *name* | *name* | The name of the protected Add-On Instruction. |
| *SignatureID* | *SignatureID* | 32-bit value based upon the current configuration of the Add-On-Instruction. |
| *SignatureTimestamp* | *SignatureTimestamp* | The time and date that the Add-On-Instruction was sealed. |
| *SafetySignatureID* | *SafetySignatureID* | Only applies to Safety Add-On-Instructions. Additional safety ID calculated online in the safety system. |
| N/A | *other_attributes* | Attributes of the Add-On Instruction that are not protected during export. |
| N/A | *encoded_data* | The protected portion of the Add-On Instruction. |
| *IsEncrypted* | *IsEncrypted* | Indicates whether the Add-On Instruction is protected with license-based Source Protection and locked. |

| | |
|---|---|
| **Important:** | When the Add-On Instruction is source-protected, the *encoded_data* information is encrypted. If you modify this encrypted information, you cannot re-import the Add-On Instruction. |

## Encoded Information elements

| L5X Item | L5K Item | Identifies |
|---|---|---|
| EncryptionKey | ENCRYPTION_KEY | Identifies the options the user has chosen for protecting and locking their content. |

## Encoded key attributes

| Attribute | Description |
|---|---|
| Name | Identifies the type of protection. |
| ID | Identifier for the key (Firm code, product code). |
| Description | Description of what the key is associated with (license name). |
| Vendor | Indicates the vendor who supplied the key. |
| PublicKey | Stores the public key that will be used for the locking of the associated object. |

## Encoded content attributes

| Attribute | Description |
|---|---|
| EncryptedType | Indicates the underlying language of the routine for this encoded content (for example, RLL or Structured text). |
| OnlineEditType | L5X only. Specify the online edit logic type (**Original**, **PendingEdits**, or **TestEdits**). This attribute is not specified if there are no edits. |

## L5X Encoded Add-On Instruction example

```
- <EncodedData EncodedType="AddOnInstructionDefinition" Name="Conveyor_Control" Revision="1.0"
    Vendor="vendor" SignatureID="AC2CCC57" SignatureTimestamp="2014-05-20T14:04:14.807Z"
    EditedDate="2014-05-20T14:04:14.807Z" SoftwareRevision="v26.00" EncryptionConfig="3">
  - <Description>
      <![CDATA[ text  ]]>
    </Description>
  - <RevisionNote>
      <![CDATA[ text  ]]>
    </RevisionNote>
  - <SignatureHistory>
    - <HistoryEntry User="RA-INT\JBieder2" Timestamp="2014-05-20T00:15:08.867Z"
        SignatureID="16#52db_eb8a">
      - <Description>
          <![CDATA[ text  ]]>
        </Description>
      </HistoryEntry>
    - <HistoryEntry User="RA-INT\JBieder2" Timestamp="2014-05-20T13:38:51.670Z"
        SignatureID="16#52db_eb8a">
      - <Description>
          <![CDATA[ text  ]]>
        </Description>
      </HistoryEntry>
    </SignatureHistory>
  - <AdditionalHelpText>
      <![CDATA[ text  ]]>
    </AdditionalHelpText>
  - <Parameters>
    - <Parameter Name="EnableIn" TagType="Base" DataType="BOOL" Usage="Input" Radix="Decimal"
        Required="false" Visible="false" ExternalAccess="Read Only">
      - <Description>
          <![CDATA[ Enable Input - System Defined Parameter  ]]>
        </Description>
      </Parameter>
    - <Parameter Name="EnableOut" TagType="Base" DataType="BOOL" Usage="Output" Radix="Decimal"
        Required="false" Visible="false" ExternalAccess="Read Only">
      - <Description>
          <![CDATA[ Enable Output - System Defined Parameter  ]]>
        </Description>
      </Parameter>
    </Parameters>
    vchdvJHwlUXY5J4s3WA7mVzsOPGvdH8a7Ab6Yu6rXA+I0n+Rbxo6WR2O8WDtpzuFv4v/DO0KGNs2T
  </EncodedData>
```

## L5K Encoded Add-On instruction example

```
ENCODED_DATA (EncodedType := ADD_ON_INSTRUCTION_DEFINITION,
              Name := "Conveyor_Control",
              Description := "This is the description",
              Revision := "1.0",
              RevisionNote := "This is a Revision Note",
              Vendor := "vendor",
              SignatureID := AC2CCC57,
              SignatureTimestamp := "2014-05-20T14:04:14.807Z",
              EditedDate := "2014-05-20T14:04:14.807Z",
              AdditionalHelpText := "This is help text",
              EncryptionConfig := 3)
      HISTORY_ENTRY  (User := RA-INT\JBieder2,
                      Timestamp := "2014-05-20T00:15:08.867Z",
                      SignatureID := 16#52db_eb8a,
                      Description := "History description")
      END_HISTORY_ENTRY
      PARAMETERS
            EnableIn : BOOL (Description := "Enable Input -
System Defined Parameter",
                                 Usage := Input,
                                 RADIX := Decimal
                                 Required := No,
                                 Visible := No,
                                 ExternalAccess := Read Only);
            EnableOut : BOOL (Description := "Enable Output -
System Defined Parameter",
                                  Usage := Output,
                                  RADIX := Decimal,
                                  Required := No,
                                  Visible := No,
                                  ExternalAccess := Read
Only);
      END_PARAMETERS
      5PC4UUeSPrD8+QMe30neT5/97J+VmK95qgOApHiZ7VpmkuGyeYVmzDm3
ceYND35YMmzC4xyFQfJYld…
END_ENCODED_DATA
```

## Add-On Instruction Guidelines

Use these Add-On Instruction guidelines with function blocks:

- If the operand is not a qualified tag or literal value, the Add-On Instruction is not verified.
- The X and Y grid locations are a relative position from the upper-left corner of the sheet. X is the horizontal position; Y is the vertical position.

## L5X unencoded AddOnInstruction definition example

This L5X file shows a definition partial export example for an L5X Add-On Instruction.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <RSLogix5000Content SchemaRevision="1.0" SoftwareRevision="26.00" TargetType="AddOnInstructionDefinition" ContainsContext="true"
    Owner="Rockwell Automation, Rockwell Automation" ExportDate="Mon Oct 16 12:57:21 2006" ExportOptions="Context">
  - <Controller Use="Context" Name="My_Controller_1">
    - <AddOnInstructionDefinitions Use="Context">
      - <AddOnInstructionDefinition Use="Target" Name="Conveyor_Control" Revision="1.0" Vendor="Rockwell" ExecutePrescan="false"
          ExecutePostscan="false" ExecuteEnableInFalse="false" CreatedDate="2005-10-24T18:50:43.227Z" CreatedBy="NA\Mills"
          EditedDate="2005-12-13T20:13:00.825Z" EditedBy="NA\Mills" SoftwareRevision="v26.00">
        - <Description>
            <![CDATA[ Starts and stops a conveyor  ]]>
          </Description>
        - <AdditionalHelpText>
          - <![CDATA[
              Use this instruction to start and stop a conveyor. Use the instruction together with a photoeye or other digital
              sensor that detects product at the entry to the conveyor.
              - If the Stop button is closed, the conveyor starts when product passes the entry sensor.
              - The conveyor stops when you press (open) the Stop button.
              - Use the Jog bit to jog the conveyor. The Jog bit overrides the Stop button.

              Automatically Turn Off the Conveyor
              You can configure the instruction to automatically turn off the conveyor if there's no product after a certain time.
              - In NoLoadTime, enter how long you want to wait for product. Enter the time in milliseconds.
              - The conveyor turns off if product doesn't show up within the NoLoadTime.
              -You must set NoLoadTime greater than 0 to automatically turn off the conveyor. Otherwise it keeps running until you press (open)

              Check for a Jam
              You can configure the instruction to automatically turn off the conveyor if product gets stuck at the entry.
              - In JamTime, enter how long to wait before signaling a jam. Enter the time in milliseconds.
              -The conveyor turns off and the Jam bit turns on if the input sensor stays on for JamTime.
              -You must set JamTime greater than 0 to signal a Jam. Otherwise the conveyor keeps running until you stop it.
              - To clear the Jam bit, turn on the JamClear bit.

              Watch for a Motor Fault
              You can also use the auxiliary contact of the conveyor's motor to make a fault happen if the motor doesn't start or stop.
              - In FaultTime, enter how long you want to wait for the contact to open or close. Enter the time in milliseconds.
              - The Fault bit turns on if the contact doesn't show that the motor started or stopped within the FaultTime.
              -You must set FaultTime greater than 0 to use the auxiliary contact. Otherwise the instruction doesn’t use the value of the auxili
              - To clear the Fault bit, turn on the ClearFault bit.

              In the LD and ST programming languages, this instruction doesn't show all its operands. Use other instructions to access the opera
              - In LD, use XIC and OTE instructions to read the value of the auxiliary contact tag and write it to the AuxContact bit.
              - In ST, use an assignment (:=) to set the AuxContact bit equal to the value of the auxiliary contact tag.
            ]]>
          </AdditionalHelpText>
        - <Parameters>
          - <Parameter Name="Stop" DataType="BOOL" Usage="Input" Radix="Decimal" Required="true" Visible="true">
            - <Description>
                <![CDATA[ Enter the tag for the stop pushbutton for the conveyor.  ]]>
              </Description>
              <DefaultData>00</DefaultData>
            </Parameter>
          + <Parameter Name="Start" DataType="BOOL" Usage="Input" Radix="Decimal" Required="true" Visible="true">
          + <Parameter Name="AuxContact" DataType="BOOL" Usage="Input" Radix="Decimal" Required="false" Visible="false">
          + <Parameter Name="Jog" DataType="BOOL" Usage="Input" Radix="Decimal" Required="false" Visible="false">
          + <Parameter Name="JamClear" DataType="BOOL" Usage="Input" Radix="Decimal" Required="false" Visible="false">
          + <Parameter Name="ClearFault" DataType="BOOL" Usage="Input" Radix="Decimal" Required="false" Visible="false">
          + <Parameter Name="Out" DataType="BOOL" Usage="Output" Radix="Decimal" Required="true" Visible="true">
          + <Parameter Name="Jam" DataType="BOOL" Usage="Output" Radix="Decimal" Required="false" Visible="true">
          + <Parameter Name="Fault" DataType="BOOL" Usage="Output" Radix="Decimal" Required="false" Visible="true">
          + <Parameter Name="JamTime" DataType="DINT" Usage="Input" Radix="Decimal" Required="false" Visible="false">
          + <Parameter Name="FaultTime" DataType="DINT" Usage="Input" Radix="Decimal" Required="false" Visible="false">
          + <Parameter Name="NoLoadTime" DataType="DINT" Usage="Input" Radix="Decimal" Required="false" Visible="false">
          </Parameters>
```

## Add-On Instruction Example, Continued

```
- <LocalTags>
  + <LocalTag Name="NoLoadTimer" DataType="TIMER">
  + <LocalTag Name="JamTimer" DataType="TIMER">
  + <LocalTag Name="Motor_Starter" DataType="Motor_Starter">
  + <LocalTag Name="OK_To_Run" DataType="BOOL" Radix="Decimal">
  + <LocalTag Name="CheckAuxContact" DataType="BOOL" Radix="Decimal">
  + <LocalTag Name="CheckJam" DataType="BOOL" Radix="Decimal">
  </LocalTags>
- <Routines>
  - <Routine Name="Logic" Type="RLL">
    - <Description>
        <![CDATA[ Runs a conveyor based on start and stop inputs ]]>
      </Description>
    - <RLLContent>
      - <Rung Number="0" Type="N">
        - <Text>
            <![CDATA[ XIC(Stop)XIO(Jam)XIO(NoLoadTimer.DN)OTE(OK_To_Run); ]]>
          </Text>
        </Rung>
      - <Rung Number="1" Type="N">
        - <Text>
            <![CDATA[ XIC(Jog)OTE(Motor_Starter.Jog); ]]>
          </Text>
        </Rung>
      + <Rung Number="2" Type="N">
      + <Rung Number="3" Type="N">
      + <Rung Number="4" Type="N">
      + <Rung Number="5" Type="N">
      + <Rung Number="6" Type="N">
      + <Rung Number="7" Type="N">
      + <Rung Number="8" Type="N">
      + <Rung Number="9" Type="N">
      </RLLContent>
    </Routine>
  </Routines>
- <Dependencies>
    <Dependency Type="AddOnInstructionDefinition" Name="Motor_Starter" />
  </Dependencies>
 </AddOnInstructionDefinition>
</AddOnInstructionDefinitions>
</Controller>
</RSLogix5000Content>
```

## L5K unencoded ADD_ON_INSTRUCTION _DEFINITION example

```
ADD_ON_INSTRUCTION_DEFINITION Valve (Description := "Simple
valve control",
            Revision := “1.0”, RevisionExtension := “B”,
            Vendor := “RaesUDICreationsUnlimited”,
ExecutePrescan := Yes,
            ExecutePostscan := No, ExecuteEnableInFalse :=
No,
            CreatedBy := “apollo\drjones”, EditedDate :=
“2005-01-05T15:24:59.188Z”,
            EditedBy := “apollo\drjones”,
            AdditionalHelpText := “My first Add-On
Instruction – how cool!”)

      PARAMETERS
            Valve_Command : BOOL (Description := “0 - Close
valve$N1 - Open valve”,
            Radix := Decimal, Required := Yes, Visible :=
Yes, DefaultData := “1”);
            Array_Parameter : REAL[5] (Type := InOut, Radix
:= Float, Required := Yes,
            Visible := Yes); Valve_Out : DINT (Type :=
Output, Radix := Decimal,
```

```
            Required := No, Visible := Yes, DefaultData :=
“0”);
            Reset : BOOL (Description := “Used by Prescan
routine to run Reset code”,
            Type := Input, Radix := Decimal, Required := No,
Visible := No,
            DefaultData := “1”);
      END_PARAMETERS
      LOCAL_TAGS
            Valve_Type : DISCRETE_2STATE (Description := “The
valve is a 2 state valve”,
            DefaultData :=
“[49,0.00000000e+000,0,0,0.00000000e+000,0.00000000e+000,

      0.00000000e+000,0.00000000e+000,0.00000000e+000,0.000000
00e+000]”);
      END_LOCAL_TAGS
      FBD_ROUTINE Logic (Description := "This UDI Logic
routine is nonsense but shows the
      format sufficiently. In fact, it does not even use the
InOut Parameter",
            SheetSize := "Letter (8.5x11in)",
SheetOrientation := Landscape)
            SHEET (Name := "")
                  D2SD_BLOCK (ID := 0, X := 200, Y := 160,
Operand := Valve_Type,
                        VisiblePins := "ProgCommand,
State0Perm, State1Perm, FB0, FB1,
                        HandFB, ProgProgReq, ProgOperReq,
ProgOverrideReq, ProgHandReq,
                        Out, Device0State, Device1State,
CommandStatus, FaultAlarm,
                        ModeAlarm, ProgOper, Override,
Hand")
                  END_D2SD_BLOCK
                  IREF (ID := 1, X := 120, Y := 100, Operand
:= Valve_Command)
                  END_IREF
                  OREF (ID := 2, X := 460, Y := 140, Operand
:= Valve_Out)
                  END_OREF
            END_SHEET
      END_FBD_ROUTINE
      ST_ROUTINE Prescan (Description := "This should run
before the Instruction does")
            '//If Reset is True - do something
            'IF (Reset) THEN
            ' //do something
            'END_IF;
            '
      END_ST_ROUTINE

END_ADD_ON_INSTRUCTION_DEFINITION
```

## L5X unencoded safety AddOnInstruction definition example

```
– <AddOnInstructionDefinition Use="Target" Name="HI_SafetyAOI" Revision="1.0"
RevisionExtension="B" Vendor="AOICreationsUnlimited" Class="Safety" ExecutePrescan="true"
ExecutePostscan="false" ExecuteEnableInFalse="false" CreatedData="2009-01-05T15:24:59.188Z"
CreatedBy="apollo\drjones" EditedDate="2009-02-25T15:05:52.042Z" EditedBy="apollo\drjones"
SoftwareRevision="V18.00" >

– <Description>
      – <![CDATA[ Simple valve control ]]>
  </Description>
– <RevisionNote>
      – <![CDATA[ Original release to library ]]>
  </RevisionNote>
– <SignatureHistory>
      – <HistoryEntry User="apollo\drjones" Timestamp="2009-01-05T15:24:59.188Z"
      SignatureID="68F42D31" >
            – <Description>
                  - <![CDATA[ My First History Entry! ]]>
              </Description>
        </HistoryEntry>
      – <HistoryEntry User="apollo\drjones" Timestamp="2009-02-03T10:24:19.760Z"
      SignatureID="C7013D42" >
            – <Description>
                  - <![CDATA[ My Second History Entry! ]]>
              </Description>
        </HistoryEntry>
      – <HistoryEntry User="apollo\drjones" Timestamp="2009-02-25T15:05:52.042Z "
      SignatureID=" F4E691A2" >
            – <Description>
                  - <![CDATA[ My Last History Entry! ]]>
              </Description>
        </HistoryEntry>
  </SignatureHistory>
– <AdditonalHelpText>
      – <![CDATA[ My first Add-On Instruction – how cool! ]]>
  </ AdditonalHelpText >
– <Parameters>
      <!-- parameters deleted for brevity-->
  </Parameters>
– <LocalTags>
      <!—local tags deleted for brevity-->
  </Localtags>
– <Routines>
      <!—routines deleted for brevity-->
  </Routines>
  </ AddOnInstructionDefinition>
```

## L5K unencoded safety ADD_ON_INSTRUCTION _DEFINITION example

```
ADD_ON_INSTRUCTION_DEFINITION HI_SafetyAOI (Description :=
"sealed safety AOI",
      Revision := “1.0”, RevisionExtension := “B”,
      RevisionNote := “Original release to library”,
      Vendor := “AOICreationsUnlimited”, Class := Safety,
ExecutePrescan := Yes,
      ExecutePostscan := No, ExecuteEnableInFalse := No,
      CreatedDate := “2009-01-05T15:24:59.188Z”, CreatedBy :=
“apollo\drjones”,
      EditedDate := “2009-02-25T15:05:52.042Z”, EditedBy :=
“apollo\drjones”,
      AdditionalHelpText := “My first HI Safety Add-On
Instruction”)

HISTORY_ENTRY (User := “apollo\drjones”,
      Timestamp := “2009-01-05T15:24:59.188Z”, SignatureID :=
68F42D31,
      Description := “My First History Entry!”)
END_ HISTORY_ENTRY

HISTORY_ENTRY (User := “apollo\drjones”,
      Timestamp := “2009-02-03T10:24:19.760Z”, SignatureID :=
C7013D42,
      Description := “My Second History Entry!”)
END_ HISTORY_ENTRY

HISTORY_ENTRY (User := “apollo\drjones”,
      Timestamp := “2009-02-25T15:05:52.042Z”, SignatureID :=
F4E691A2,
      Description := “My Last History Entry!”)
END_ HISTORY_ENTRY
(* PARAMETERS, LOCAL_TAGS, and ROUTINE blocks deleted for
brevity *)
END_ADD_ON_INSTRUCTION_DEFINITION
```

# Define a tag component

## Introduction

This chapter explains the overall structure of the tag component.

## Tag component

The tag component defines the tags associated with the logic you selected or within the program you selected. Within a tag list in the L5K format, message and motion tags must follow all non-motion tags, and axis tags must follow motion group tag. Tags may appear in any order in the L5X format.

| | |
|---|---|
| **Important:** | For detailed information about atomic and structure tags and their supported attributes and ranges, see the Logix5000 Controller Common Procedures Programming Manual, publication 1756-PM001. |

## L5X tag structure

```
<Tag [Tag_Attributes]>
      <ConsumeInfo [Consume_Attributes]/>
      <ProduceInfo [Produce_Attributes]/>
      <Description>
            <![CDATA[ text ]]>
      </Description>
      <Comments>
            <Comment Operand="specifier">
                  <![CDATA[ comment_text ]]>
            </Comment>
      </Comments>
      <EngineeringUnits>
            <EngineeringUnit Operand="specifier">
                  <![CDATA[ engineering_unit_text ]]>
            </EngineeringUnit>
      </EngineeringUnits>
      <Mins>
            <Min Operand="specifier"> min_value </Min>
      </Mins>
      <Maxes>
            <Max Operand="specifier"> max_value </Max>
      </Maxes>
      <State0s>
            <State0 Operand="specifier">
                  <![CDATA[ state0_text ]]>
            </State0>
      </State0s>
```

```
            <State1s>
                  <State1 Operand="specifier">
                        <![CDATA[ state1_text ]]>
                  </State1>
            </State1s>
            <Data>
                  value
            </Data>
            <ForceData>
                  value
            </ForceData
      </Tag>
```

## L5K TAG structure

```
      <tag_name> [OF alias] : <type["x,y,z"]>
            [(Description := "text",
            Comment := "text",
            Comment := "text",
            EngineeringUnit := "text",
            Max := value,
            Min := value,
            State0 := "text",
            State1 := "text",
            Tag_Attributes,
            Produce_Attributes,
            Consume_Attributes)]
            [, <tag_force_data>] := value;
```

## Tag elements

| L5X Item | L5K Item | Description |
|---|---|---|
| N/A | *tag_name* | The name of the tag.<br><br>In L5X, use a Name attribute on the <Tag> element. |
| N/A | *alias* | Name of the base tag that the alias tag references.<br><br>Specify *tag*<*specifier*><br>Where the *specifier* is a bit (.*bitnumber*), array element ([*element*]), or structure member (.*membername*) of the tag.<br><br>In L5X, use an AliasFor attribute on the <Tag> element. |
| N/A | *x, y, z* | The number of elements within the array dimension.<br>For example [5, 10, 2].<br><br>In an L5K array tag, there cannot be any white space between the type and array definition. There must be a space between the tag name and the colon and another space between that same colon and the type name.<br><br>In L5X, use a Dimension attribute on the <Tag> element. |
| ConsumeInfo | N/A | Identifies a consumed tag and provides the tag attributes.<br><br>In L5K, specify Consume_Attributes attributes. |

| L5X Item | L5K Item | Description |
|---|---|---|
| ProduceInfo | N/A | Identifies a produced tag and provides the tag attributes.<br><br>In L5K, specify Produce_Attributes attributes. |
| Description | Description | User information about the tag. |
| Comment | Comment | Provide information about a tag component.<br>Can specify Comment<*specifier*><br>Where the *specifier* is:<br>*.bitnumber* for a bit in the tag.<br>[*element*] for an array element of the tag.<br>*.membername* for a structure member of the tag.<br><br>There can be multiple comment elements. |
| Data | *value* | Tag data.<br><br>In L5X, this element can also contain additional attributes for other tag types. |
| EngineeringUnit | EngineeringUnit | (optional) User-specified description of what the unit of the value is, in feet, gallons, or kilos.<br><br>Can specify EngineeringUnit <*specifier*><br><br>Where the *specifier* is:<br>*.bitnumber* - for a bit in the tag<br>*[element]* - for an array element of the tag<br>*.membername* - for a structure member of the tag<br><br>There can be multiple engineering unit elements. |
| Max | Max | (optional) User-specified maximum value about qualified sub-regions of the parameter. Only valid for the sub-regions of the parameter, which is a non-Boolean atomic datatypes.<br><br>Can specify EngineeringUnit <*specifier*><br><br>Where the *specifier* is:<br>*.bitnumber* - for a bit in the tag<br>*[element]* - for an array element of the tag<br>*.membername* - for a structure member of the tag<br><br>There can be multiple max elements. |
| Min | Min | (optional) User-specified minimum value about qualified sub-regions of the parameter. Only valid for the sub-regions of the parameter, which is a non-Boolean atomic datatypes.<br><br>Can specify EngineeringUnit <*specifier*><br><br>Where the *specifier* is:<br>*.bitnumber* - for a bit in the tag<br>*[element]* - for an array element of the tag<br>*.membername* - for a structure member of the tag<br><br>There can be multiple min elements. |

| L5X Item | L5K Item | Description |
|---|---|---|
| State0 | State0 | (optional) For Boolean parameters or sub-regions only. User-specified description of what the Zero state of the Boolean value is.<br><br>Can specify EngineeringUnit <*specifier*><br><br>Where the *specifier* is:<br>*.bitnumber* - for a bit in the tag<br>*[element]* - for an array element of the tag<br>*.membername* - for a structure member of the tag<br><br>There can be multiple stat0 elements. |
| State1 | State1 | (optional) For Boolean parameters (or sub-regions) only. User-specified description of what the One state of the Boolean value is.<br><br>Can specify EngineeringUnit <*specifier*><br><br>Where the *specifier* is:<br>*.bitnumber* - for a bit in the tag<br>*[element]* - for an array element of the tag<br>*.membername* - for a structure member of the tag<br><br>There can be multiple state1 elements. |
| ForceData | *tag_force_data* | Tag force data. |

## Tag attributes

| Attribute | Description |
|---|---|
| Name | The name of the tag.<br><br>In L5K, the name is an element of the statement. |
| TagType | L5X only. Type **Base**, **Alias**, **Produce**, or **Consumed**. |
| DataType | Type of tag.<br>Atomic types: BOOL, SINT, INT, DINT, LINT, REAL<br>String types: STRING<br>Predefined types such as: AXIS_CONSUMED, AXIS_GENERIC_DRIVE, AXIS_SERVO, AXIS_SERVO_DRIVE, AXIS_VIRTUAL, CAM, CAM_PROFILE, CONTROL, COORDINATE_SYSTEM, COUNTER, MESSAGE, MOTION_GROUP<br>Equipment phase types: PHASE, PHASE_INSTRUCTION<br>Safety types: CONNECTION_STATUS and unique types for each safety instruction<br>Function block types: unique type for each function block<br>Sequential function chart: SFC_ACTION, SFC_STEP, SFC_STOP<br>User-defined data types.<br>Add-On Instruction defined data types.<br>Module-defined data types. |

| Attribute | Description |
|---|---|
| AliasFor | L5X only. Name of the base tag that the alias tag references.<br><br>Specify *tag<specifier>*<br>Where the *specifier* is a bit (*.bitnumber*), array element ([*element*]), or structure member (*.membername*) of the tag, or any combination such as *[7].Input.0*. |
| Dimensions | L5X only. The number of elements within the array dimension.<br>For example [5, 10, 2]. |
| Class | Specify the class of the tag. This attribute applies only to safety controller projects. Type **Standard** or **Safety**. |
| Radix | Specify the display style as **decimal**, **hex**, **octal**, **binary**, **exponential**, **float**, **ASCII**, or **date/time**. (LINT only). |
| PLCMappingFile | If this tag is mapped to a PLC controller, specify the file number, which can be any positive number.<br><br>For L5X, this attribute is on the <ProduceInfo> element. |
| PLC2Mapping | If this tag is mapped to a PLC-2 file, set this attribute to 1. If this tag is not mapped to a PLC-2 file, set this attribute to 0.<br><br>For L5X, this attribute is on the <ProduceInfo> element. |
| ProgrammaticallySend EventTrigger | If the project programmatically sends an event trigger, set this attribute to **1**. Otherwise, set this attribute to **0**.<br><br>For L5X, this attribute is on the <ProduceInfo> or <ConsumeInfo> element. |
| Unicast | Allow connections to be unidirectional, rather than bidirectional. Type **Yes** or **No**. |
| UnicastPermitted | Specify when unicast connections can be received. Type **Yes** or **No**.<br><br>For L5X, this attribute is on the <ProduceInfo> element. |
| Usage | • Specify how an Equipment Phase program uses a tag. This attribute applies only to tags that are program-scoped to an Equipment Phase program. Type **Input**, **Output**, or **Normal**.<br>• Specify how a parameter is used. Type **Input**, **Output**, **InOut**, or **Public**. |
| Sequencing | Specify if the parameter can be seen by the FactoryTalk Batch Server. Only Input and Output parameters can be used by a sequence. Usage is **Input** or **Output**. |
| ExternalAccess | Specify the external access, outside of the controller, to the tag. Specify **Read/Write**, **Read Only**, or **None**. |
| Max | (optional) User specified maximum value for the tag. Only valid for tag with non-Boolean atomic datatype. |
| Min | (optional) User specified minimum value for the tag. Only valid for tag with non-Boolean atomic datatype. |
| Constant | Specify whether the tag value is a constant value or it can change. For L5K, specify **yes** for a constant value or **no** for a dynamic value. For L5X, specify **true** or **false**. |
| PermissionSet | Name of the set of permissions, configured in FactoryTalk Security, to apply to this object. |
| TrackingGroups | The group of tracked objects to which this item belongs. Components can be marked for tracking to determine whether they have been changed. Version 30 of the Logix Designer application supports only one tracking group. |

## Produced tag attributes

A produced tag has these attributes, and those for a standard tag.

| Attribute | Description |
|---|---|
| ProduceCount | Specify the number of consumers allowed with any positive number.<br><br>In L5X, this attribute is on the <ProduceInfo> element. |

| Attribute | Description |
|---|---|
| MinimumRPI | Specify the smallest and fastest packet interval at which consumers may consume data from the tag (0.196...536870.911 and 1...536870.911 for CompactLogix). |
| MaximumRPI | Specify the largest and slowest packet interval at which consumers may consume data from the tag (0.196...536870.911). |
| DefaultRPI | Specify a default RPI that the produced tag provides value to consumers that attempt to connect with an out-of-range RPI (0.196...536870.911). |

## Consumed tag attributes

A consumed tag also has these attributes, in addition to those for a standard tag.

**Important:** If consumed information is provided on an alias tag, the alias tag is converted to a base tag before it can consume data.

| Attribute | Description |
|---|---|
| Producer | If the controller consumes this tag, specify the name of the remote controller that produces this tag. You must also specify RemoteTag and RPI attributes.<br><br>In L5X, this attribute is on the <ConsumeInfo> element. |
| RemoteTag | If the controller consumes this tag from a controller that supports tag names, specify the name of the tag on the remote controller. You must also specify Producer and RPI attributes.<br><br>In L5X, this attribute is on the <ConsumeInfo> element. |
| RemoteFile | If the controller consumes this tag from a PLC-5 controller, specify the PLC-5 file number, as any positive number, on the PLC-5 controller. You must also specify Producer and RPI attributes.<br><br>In L5X, this attribute is on the <ConsumeInfo> element. |
| RPI | If the controller consumes this tag, specify the RPI value in milliseconds, as any positive number. You must also specify Producer and RemoteTag attributes.<br><br>In L5X, this attribute is on the <ConsumeInfo> element. |

# ALARM_ANALOG tag

## L5X tag structure for ALARM_ANALOG tag

In an.L5X file, tag attributes for an ALARM_ANALOG tag are in the Data element.

```
<Data Format="Alarm">
      <AlarmAnalogParameters [Alarm_Analog_Attributes]/>
      <AlarmConfig>
            messages
            alarm_class
            HMI_command
      </AlarmConfig>
</Data>
```

## L5K tag structure for ALARM_ANALOG tag

In an.L5K file, tag attributes for an ALARM_ANALOG tag are in the tag statement.

```
<tag_name> : <type>
      [(Alarm_Analog_Attributes)]
```

## ALARM_ANALOG tag attributes

| Attribute | Description |
|---|---|
| EnableIn | Specify whether to enable the alarm tag. Type **0** to disable the tag; type **1** to enable the tag. |
| InFault | Specify the quality of the input fault data. Type **1** for bad quality; type **0** for good quality. |
| HHEnabled | Specify the whether the alarm monitors for a high-high limit. Type **1** to enable; type **0** to disable. |
| HEnabled | Specify the whether the alarms monitors for a high limit. Type **1** to enable; type **0** to disable. |
| LEnabled | Specify the whether the alarm monitors for a low limit. Type **1** to enable; type **0** to disable. |
| LLEnabled | Specify the whether the alarms monitors for a low-low limit. Type **1** to enable; type **0** to disable. |
| AckRequired | Specify whether the alarms requires acknowledgment. Type **1** to enable; type **0** to disable. |
| ProgAckAll | Specify whether the program acknowledges all alarm conditions. Type **1** to enable; type **0** to disable. |
| OperAckAll | Specify whether an operator acknowledges all alarm conditions. Type **1** to enable; type **0** to disable. |
| HHProgAck | Specify whether the program acknowledges a high-high condition. Type **1** to enable; type **0** to disable. |
| HHOperAck | Specify whether an operator acknowledges a high-high condition. Type **1** to enable; type **0** to disable. |
| HProgAck | Specify whether the program acknowledges a high condition. Type **1** to enable; type **0** to disable. |
| HOperAck | Specify whether an operator acknowledges a high condition. Type **1** to enable; type **0** to disable. |
| LProgAck | Specify whether the program acknowledges a low condition. Type **1** to enable; type **0** to disable. |
| LOperAck | Specify whether an operator acknowledges a low condition. Type **1** to enable; type **0** to disable. |
| LLProgAck | Specify whether the program acknowledges a low-low condition. Type **1** to enable; type **0** to disable. |
| LLOperAck | Specify whether an operator acknowledges a low-low condition. Type **1** for enabled; type **0** for disabled. |
| HHOperShelve | Set by the operator to shelve a high-high condition. |
| HOperShelve | Set by the operator to shelve a high condition. |
| LOperShelve | Set by the operator to shelve a low condition. |
| LLOperShelve | Set by the operator to shelve a low-low condition. |
| ROCPosProgAck | Specify whether the program acknowledges a positive (increasing), rate-of-change condition. Type **1** to enable; type **0** to disable. |
| ROCPosOperAck | Specify whether an operator acknowledges a positive (increasing), rate-of-change condition. Type **1** to enable; type **0** to disable. |
| ROCPNegProgAck | Specify whether the program acknowledges a negative (decreasing), rate-of-change condition. Type **1** to enable; type **0** to disable. |
| ROCPNegOperAck | Specify whether an operator acknowledges a negative (decreasing), rate-of-change condition. Type **1** to enable; type **0** to disable. |
| ROCPosOperShelve | Set by the operator to shelve a positive rate-of-change condition. |
| ROCNegOperShelve | Set by the operator to shelve a negative rate-of-change condition. |
| ProgSuppress | Specify whether the program can suppress an alarm. Type **1** to enable; type **0** to disable. |
| OperSuppress | Specify whether an operator can suppress an alarm. Type **1** to enable; type **0** to disable. |
| ProgUnsuppress | Specify whether the program can unsuppress an alarm. Type **1** to enable; type **0** to disable. |
| OperUnsuppress | Specify whether an operator can unsuppress an alarm. Type **1** to enable; type **0** to disable. |
| ProgDisable | Specify whether the program disables an alarm. Type **1** to enable; type **0** to disable. |
| OperDisable | Specify whether an operator disables an alarm. Type **1** to enable; type **0** to disable. |

| Attribute | Description |
|---|---|
| ProgEnable | Specify whether the program enables an alarm. Type **1** to enable; Type **0** to disable. |
| OperEnable | Specify whether an operator enables an alarm. Type **1** to enable; type **0** to disable. |
| ProgUnshelveAll | Set by the user program to unshelve all conditions on this alarm. |
| AlarmCountReset | Specify whether to reset the alarm count. Type **1** to reset; type **0** to not reset. |
| In | Specify the analog input (REAL) to the alarm. |
| HHLimit | Specify the high-high limit (REAL) for the alarm condition. |
| HHSeverity | Specify the severity (1...500) of a high-high alarm condition. |
| HLimit | Specify the high limit (REAL) for the alarm condition. |
| HSeverity | Specify the severity (1...500) of a high alarm condition. |
| LLimit | Specify the low limit (REAL) for the alarm condition. |
| LSeverity | Specify the severity (1...500) of a low alarm condition. |
| LLLimit | Specify the low-low limit (REAL) for the alarm condition. |
| LLSeverity | Specify the severity (1...500) of a low-low alarm condition. |
| HHOperUnshelve | Set by the operator to unshelve a high-high condition. |
| HOperUnshelve | Set by the operator to unshelve a high condition. |
| LOperUnshelve | Set by the operator to unshelve a low condition. |
| LLOperUnshelve | Set by the operator to unshelve a low-low condition. |
| MinDurationPRE | Specify the minimum time (DINT) an alarm condition to remain true for the alarm to be considered active. |
| HHMinDurationEnable | Set to enable minimum duration timer when detecting the high-high condition. |
| HMinDurationEnable | Set to enable minimum duration timer when detecting the high condition. |
| LMinDurationEnable | Set to enable minimum duration timer when detecting the low condition. |
| LLMinDurationEnable | Set to enable minimum duration timer when detecting the low-low condition. |
| Deadband | Specify the deadband (REAL) for the high-high, high, low, and low-low levels. |
| ROCPosLimit | Specify the positive rate-of-change limit (REAL) for the alarm condition. |
| ROCPosSeverity | Specify the severity (1...500) of a positive rate-of-change alarm condition. |
| ROCNegLimit | Specify the negative rate-of-change limit (REAL) for the alarm condition. |
| ROCNegSeverity | Specify the severity (1...500) of a negative rate-of-change alarm condition. |
| ROCPeriod | Specify the time period (seconds) to evaluate rate-of-change conditions. |
| ROCPosOperUnshelve | Set by the operator to unshelve a positive rate-of-change condition. |
| ROCNegOperUnshelve | Set by the operator to unshelve a negative rate-of-change condition. |
| ShelveDuration | Time duration for which a shelved alarm will be shelved, between 1 and *MaxShelveDuration*. |
| MaxShelveDuration | Maximum time duration for which an alarm can be shelved. |
| AssocTag1 | Specify a tag associated with the alarm. |
| AssocTag2 | Specify a tag associated with the alarm. |
| AssocTag3 | Specify a tag associated with the alarm. |
| AssocTag4 | Specify a tag associated with the alarm. |
| AlarmClass | Specify an alarm class for the alarm.<br><br>In L5X, use an AlarmClass element in the Data Element. |

| Attribute | Description |
|---|---|
| HMICmd | Specify a command string for the HMI.<br><br>In L5X, use an HMICommand element in the Data Element. |

# ALARM_DIGITAL tag

## L5X tag structure for ALARM_DIGITAL tag

In an.L5X file, tag attributes for an ALARM_DIGITAL tag are in the Data element.

```
<Data Format="Alarm">
      <AlarmDigitalParameters [Alarm_Digital_Attributes]/>
      <AlarmConfig>
            message
            alarm_class
            HMI_command
      </AlarmConfig>
</Data>
```

## L5K tag structure for an ALARM_DIGITAL tag

In an.L5K file, tag attributes for an ALARM_DIGITAL tag are in the tag statement.

```
<tag_name> : <type>
      [(Alarm_Digital_Attributes)] := value;
```

## ALARM_DIGITAL tag attributes

| Attribute | Description |
|---|---|
| EnableIn | Specify whether to enable the alarm tag. Type **1** to enable the tag; type **0** to disable the tag |
| In | Specify the analog input to the alarm. |
| InFault | Specify the quality of the input fault data. Type **1** for bad quality; type **0** for good quality. |
| Condition | Specify whether the alarm condition exists. Type **1** for yes; type **0** for no. |
| AckRequired | Specify whether the alarms require acknowledgment. Type **1** to enable; type 0 to disable. |
| Latched | Specify whether the alarm output is latched. Type **1** for yes; type **0** for no. |
| ProgAck | Specify whether the program acknowledges the alarm condition. Type **1** to enable; type **0** to disable. |
| OperAck | Specify whether an operator acknowledges the alarm condition. Type **1** to enable; type **0** to disable. |
| ProgReset | Specify whether the program resets the alarm condition. Type **1** to enable; type **0** to disable. |
| OperReset | Specify whether an operator resets the alarm condition. Type **1** to enable; type **0** to disable. |
| ProgSuppress | Specify whether the program can suppress an alarm. Type **1** to enable; type **0** to disable. |
| OperSuppress | Specify whether an operator can suppress an alarm. Type **1** to enable; type **0** to disable. |
| ProgUnsuppress | Specify whether the program can unsuppress an alarm. Type **1** to enable; type **0** to disable. |
| OperUnsuppress | Specify whether an operator can unsuppress an alarm. Type **1** to enable; type **0** to disable. |
| ProgDisable | Specify whether the program disables an alarm. Type **1** to enable; type **0** to disable. |
| OperDisable | Specify whether an operator disables an alarm. Type **1** to enable; type **0** to disable. |
| ProgEnable | Specify whether the program enables an alarm. Type **1** to enable; type **0** to disable. |
| OperEnable | Specify whether an operator enables an alarm. Type **1** to enable; type **0** to disable. |

| Attribute | Description |
|---|---|
| AlarmCountReset | Specify whether to reset the alarm count. Type **1** to reset; type **0** to not reset. |
| UseProgTime | Specify how to timestamp alarm events. Type **1** for programmatic timestamp; type **0** for controller timestamp. |
| ProgTime | Specify the programmatic timestamp (LINT). |
| Severity | Specify the severity (1...500) of the alarm condition. |
| MinDurationPRE | Specify the minimum time (DINT) an alarm condition remains true for the alarm to be considered active. |
| OperShelve | Set by the operator interface to shelve the alarm. |
| ProgUnshelve | Set by the user program to unshelve the alarm. |
| OperUnshelve | Set by the operator interface to unshelve the alarm. |
| ShelveDuration | Time duration for which a shelved alarm is shelved, between 1 and *MaxShelveDuration*. |
| MaxShelveDuration | Maximum time duration that an alarm is shelved. |
| AssocTag1 | Specify a tag associated with the alarm. |
| AssocTag2 | Specify a tag associated with the alarm. |
| AssocTag3 | Specify a tag associated with the alarm. |
| AssocTag4 | Specify a tag associated with the alarm. |
| AlarmClass | Specify an alarm class for the alarm.<br><br>In L5X, use an AlarmClass element in the Data Element. |
| HMICmd | Specify a command string for the HMI.<br><br>In L5X, use an HMICommand element in the Data Element. |

## L5X AlarmConfig structure

In an.L5X file, the AlarmConfig element contains the alarm message text alarm class, and HMI command.

```
<AlarmConfig>
       <Messages>
              <Message Type="type">
                     <Text Lang="language">
                            <![CDATA[ message_text ]]>
                     </Text>
              </Message>
       /Messages>
       <AlarmClass>
              <![CDATA[ class_text ]]>
       </AlarmClass>
       <HMICommand>
              <![CDATA[ command_text ]]>
       </HMICommand>
</AlarmConfig>
```

## AlarmConfig elements

| L5X Item | Description |
|---|---|
| Message | Each message element contains a separate message. Specify a Type attribute for the analog alarm type.<br>**Specify** **For**<br>HH high-high alarm<br>H high alarm<br>L low alarm<br>LL low-low alarm<br>POS rate-of-change positive alarm<br>NEG rate-of change negative alarm |
| Text | The text of the message. Specify a Lang attribute for the language: **EN-US** (United States English), **DE** (Germany German), **ES** (Spain Spanish), **FR** (France French), **IT** (Italian), **PT** (Brazil Portuguese), **JA** (Japanese), **KO** (Korean), **ZH** (Chinese). |
| AlarmClass | Specify an alarm class for the alarm. |
| HMICommand | Specify a command string for the HMI. |

## L5K ANALOG_ALARM message structure

In an.L5K file, the ALMMSG statement contains the alarm message text.

`ALMMSG.<alarm_type>:<language>:=<"message_text">`

## ALMMSG elements

| L5K Item | Description |
|---|---|
| *alarm_type* | For an analog alarm type.<br>**Specify** **For**<br>HH high-high alarm<br>H high alarm<br>L low alarm<br>LL low-low alarm<br>POS rate-of-change positive alarm<br>NEG rate-of change negative alarm<br><br>Specify AM for a digital alarm. |
| *language* | Languages: **EN-US** (United States English), **DE** (Germany German), **ES** (Spain Spanish), **FR** (France French), **IT** (Italian), **PT** (Brazil Portuguese), **JA** (Japanese), **KO** (Korean), **ZH** (Chinese). |
| *message_text* | Alarm message.<br>Enclose the message in double quotes (" "). This is a unicode string. |

# MESSAGE tag

Message tag structure is explained in these pages.

## L5X message structure

In an.L5X file, message attributes are in the Data element.

```
<Data Format="Message">
      <MessageParameters [Message_Attributes]/>
</Data>
```

## L5K MESSAGE structure

In an.L5K file, message attributes are in the tag statement.

```
<tag_name> : <type>
      [(Message_Attributes)];
```

## Message tag attributes

| Attribute | Description |
|---|---|
| MessageType | Type **Block Transfer Read**, **Block Transfer Write**, **CIP Data Table Read**, **CIP Data Table Write**, **CIP Generic**, **PLC2 Unprotected Read**, **PLC2 Unprotected Write**, **PLC3 Typed Read**, **PLC3 Typed Write**, **PLC3 Word Range Read**, **PLC3 Word Range Write**, **PLC5 Typed Read**, **PLC5 Typed Write**, **PLC5 Word Range Read**, **PLC5 Word Range Write**, **SERCOS IDN Read**, **SERCOS IDN Write**, **SLC Typed Read**, **SLC Typed Write**, **Unconfigured**, or **Module Reconfigure**. |
| RemoteElement | Specify the address or tag name of the element in the remote device. This is the source element of a read instruction or the destination element of a write instruction. |
| RequestedLength | Specify the number of elements to be transferred (0...32,767). |
| ConnectedFlag | Specify whether the CIP generic message requires a connection or not. Type **1** for connected, or **0** for not connected. |
| ConnectionPath | Specify the connection path to the other device. |
| CommTypeCode | Specify the type of communication method.<br>**Type:** — **For this communication method:**<br>0 — CIP (most messages use CIP communications)<br>1 — DH+<br>2 — CIP with source ID<br>3 — block transfer via universal remote I/O<br>4 — block transfer via ControlNet |
| ServiceCode | If the message type is CIP Generic, specify the service code (0...32,767 hexadecimal). |
| ObjectType | If the message type is CIP Generic, specify the object type (0...32,767 hexadecimal). The ObjectType attribute is the same as the Class field on the MSG configuration dialog box. |
| TargetObject | If the message type is CIP Generic, specify the target object (0...32,767 decimal). The TargetObject attribute is the same as the Instance field on the MSG configuration dialog box. |
| AttributeNumber | If the message type is CIP Generic, specify the attribute number (0...65,535 hexadecimal). |
| Channel | For a DH+ or block transfer message, specify the channel. Type **A** or **B**. |
| SourceLink | If the communication method uses DH+, specify the source link (0...199). |
| DestinationLink | If the communication method uses DH+, specify the destination link (0...199). |
| DestinationNode | If the communication method uses DH+, specify the destination node number (0...77 octal). |
| Rack | For a DH+ or block transfer message, type the rack number (0...77 octal) of the target device. |
| Group | For a DH+ or block transfer message, type the group number (0...7) of the target device. |
| Slot | For a DH+ or block transfer message, type the slot number (0...15) of the target device. |

| Attribute | Description |
|---|---|
| LocalIndex | Specify the index into the local element, typically 0. |
| RemoteIndex | Specify the index into the remote element, typically 0. |
| LocalElement | Specify the tag name of the element in the local controller. This is the destination element of a read instruction or the source element of a write instruction. |
| DestinationTag | Specify the tag name of the destination element. |
| CacheConnections | If the message is to cache connections, type **TRUE**. If the message is not to cache connections, type **FALSE**. |
| LargePacketUsage | CIP Generic type messages with a cached connection can be configured to use either a standard or large size packet.<br>**True** makes use of the large packet size. |

## AXIS_CIP_DRIVE, AXIS_CONSUMED, AXIS_GENERIC_DRIVE, AXIS_SERVO, AXIS_SERVO_DRIVE, or AXIS_VIRTUAL Tag

Each of the axis tags has this structure.

### L5X axis structure

In an.L5X file, axis attributes are in the Data element.

```
<Data Format="Axis">
      <AxisParameters [Axis_Attributes]/>
</Data>
```

### L5K AXIS TAG structure

In an.L5K file, axis attributes are in the tag statement.

```
<tag_name> : <type>
      [(Axis_Attributes)];
```

### Axis tag attributes

These attributes are for the AXIS_CONSUMED, AXIS_GENERIC_DRIVE, AXIS_SERVO, AXIS_SERVO_DRIVE, and AXIS_VIRTUAL tags. For information about attributes for AXIS_CIP_DRIVE tags, see the *Integrated Motion on the Ethernet/IP Network Configuration and Startup User Manual* publication MOTION-UM003.

| Attribute | Description |
|---|---|
| MotionGroup | Type the name of the associated motion group, or type **NA**.<br><br>For L5X, if there is no Motion group, then the attribute is not present. |
| MotionModule | Type the name of the associated motion module, or type **NA**.<br><br>For L5X, if there is no Motion module, then the attribute is not present. |
| MotorCatalogNumber | Specify the catalog number of the motor that this axis is connected or type **NONE**. |

| Attribute | Description |
|---|---|
| RotationalPosResolution | Specify the number of counts per motor revolution (1...[232-1]). |
| ConversionConstant | Specify the number of feedback counts per position unit. Type a real number from 1.0...1.0e9. |
| OutputCamExecutionTargets | Specify the number of output cam execution targets (any positive number). |
| AxisState | Type **Axis-Ready**, **Direct Drive Control**, **Servo Control**, **Axis Faulted**, or **Axis Shutdown**. |
| PositionUnits | Specify user-defined engineering units, rather than feedback units. |
| AverageVelocityTimebase | Specify the time in seconds for calculating the average velocity of the axis (any positive number). |
| RotaryAxis | Specify the positioning mode for an axis. Type **Rotary** or **Linear**. |
| PositionUnwind | For a rotary axis, specify the distance, in feedback counts, used to perform electronic unwind (any positive number). |
| HomeMode | Specify the homing mode. Type **Passive**, **Active**, or **Absolute**. |
| HomeDirection | For active homing sequences, except for the immediate sequence type, specify the desired homing direction. Type **Uni-directional Forward**, **Bi-directional Forward**, **Uni-directional Reverse**, or **Bi-directional Reverse**. |
| HomeSequence | Specify the event that will cause the home position to be set. Type **Immediate**, **Switch**, **Marker**, **Switch-Marker**, **Torque Level**, or **Torque Level-Marker**. |
| HomeConfigurationBits | Specify the home configuration bits. Type a hexadecimal number. |
| HomePosition | Specify the desired absolute position, in positioning units, for the axis after the homing sequence is complete (any positive number). |
| HomeOffset | Specify the desired offset (any positive number) in position units the axis is to move, upon completion of the homing sequence, to reach the home position. In most cases, this value will be zero. |
| HomeSpeed | Specify the speed of the jog profile used in the first leg of the homing sequence (any positive number). The homing speed should be less than the maximum speed, and greater than zero. |
| HomeReturnSpeed | Specify speed of the jog profile used in the return leg(s) of an active homing sequence (any positive number). The return speed should be less than the maximum speed, and greater than zero. |
| MaximumSpeed | Specify the maximum speed (any positive number). |
| MaximumAcceleration | Specify the maximum acceleration rate of the axis in position units/second (any positive number). |
| MaximumDeceleration | Specify the maximum deceleration rate of the axis in position units/second (any positive number). |
| ProgrammedStopMode | Specify how a specific axis stops when the controller changes mode, or a motion group stop (MGS) instruction is executed. Type **Fast Disable**, **Fast Stop**, **Fast Shutdown**, **Hard Disable**, or **Hard Shutdown**. |
| MasterInputConfigurationBits | Specify the master input configuration bits. Type a hexadecimal number. |
| MasterPositionFilterBandwidth | Specify the bandwidth in Hertz of the master position filter. |
| AxisType | Specify the intended use of the axis. Type **Servo** or **Feedback Only**. |
| ServoLoopConfiguration | Specify the configuration of the loop. Type **Custom**, **Position Servo**, **Aux Position Servo**, **Dual Position Servo**, **Aux Command Servo, Dual Command Servo, Velocity Servo**, or **Torque Servo**. |
| ExternalDriveType | Specify the type of external drive.<br>**Specify:** **To:**<br>0 Torque servo<br>1 Velocity servo<br>2 Hydraulic servo |
| FaultConfigurationBits | Specify the fault configuration bits. Type a hexadecimal number. |

| Attribute | Description |
|---|---|
| AxisInfoSelect1 | Specify an axis attribute to transmit, and the actual position data, to the controller.<br>The options include:<br>• <none><br>• Position Command<br>• Position Feedback<br>• Aux Position Feedback<br>• Position Error<br>• Position Int.<br>• Error<br>• Velocity Command<br>• Velocity Feedback<br>• Velocity Error<br>• Velocity Int. Error<br>• Accel. Command<br>• Accel. Feedback<br>• Servo Output Level<br>• Marker Distance<br>• Torque Command<br>• Torque Feedback<br>• Positive Dynamic Torque Limit<br>• Negative Dynamic Torque Limit<br>• Motor Capacity, Drive Capacity<br>• Power Capacity<br>• Bus Regulator Capacity<br>• Motor Electrical Angle<br>• Torque Limit Source<br>• DC Bus Voltage<br>• Absolute Offset<br>• Analog Input 1<br>• Analog Input 2<br>• Guard Status<br>• Guard Faults |

| Attribute | Description |
|---|---|
| AxisInfoSelect2 | Specify a second axis attribute to transmit, and the actual position data, to the controller.<br>The options include:<br>• <none><br>• Position Command<br>• Position Feedback<br>• Aux Position Feedback<br>• Position Error<br>• Position Int. Error<br>• Velocity Command<br>• Velocity Feedback<br>• Velocity Error<br>• Velocity Int. Error<br>• Accel. Command<br>• Accel. Feedback<br>• Servo Output Level<br>• Marker Distance<br>• Torque Command<br>• Torque Feedback<br>• Positive Dynamic Torque Limit<br>• Negative Dynamic Torque Limit<br>• Motor Capacity<br>• Drive Capacity<br>• Power Capacity<br>• Bus Regulator Capacity<br>• Motor Electrical Angle<br>• Torque Limit Source<br>• DC Bus Voltage<br>• Absolute Offset<br>• Analog Input 1 or Analog Input 2<br>• Absolute Offset<br>• Analog Input 1<br>• Analog Input 2<br>• Guard Status<br>• Guard Faults. |
| LDTType | Specify the LDT device type. Type **PWM**, **Start/Stop Rising**, or **Start/Stop Falling**. |
| LDTRecirculations | Only use this field if you specified PWM for LDTType. Specify the number of recirculations that the transducer is configured for so the 1756-HYD02 module knows how the LDT is configured. |
| LDTCalibrationConstant | Specify the calibration constant (also called gradient on some LDTs). This number is engraved on each LDT by the manufacturer. It specifies the characteristics of that individual transducer. |
| LDTCalibrationConstantUnits | Specify the units of the calibration constant. Type **us/in** or **m/s**. |
| LDTScaling | Define the relationship between the unit of measurement of the transducer and the system. This is necessary for calculating the conversion constant. The LDT length is used with the number of recirculations to calculate the minimum servo update period. |
| LDTScalingUnits | Specify the units of scaling. Type **us/in** or **m/s**. |

| Attribute | Description |
|---|---|
| LDTLength | Specify the length of the LDT. |
| LDTLengthUnits | Specify the units of length. Type **us/in** or **m/s**. |
| SSICodeType | Specify the encoding on the data sent from an SSI transducer. Type **Binary** or **Grey**. |
| SSIDataLength | Specify the data length (8...32 bits) of the SSI transducer. The default value is 13. |
| SSIClockFrequency | Specify the SSI clock frequency (in kHz). Valid values are 208 (default) or 650. |
| AbsoluteFeedbackEnable | Specify whether to enable absolute feedback. Type **1** to enable absolute feedback. Otherwise, type **0**. Absolute feedback is always enabled for LDT. |
| AbsoluteFeedbackOffset | Specify the absolute offset that is used to place the machine zero point at the desired location relative to the zero point of the LDT. |
| ServoFeedbackType | Specify the type of feedback device. Type **LDT** (linear displacement transducer),<br>**AQB** (A quadrature B), or **SSI** (synchronous serial interface) |
| ServoPolarityBits | Specify the servo polarity bits. Type a hexadecimal number. |
| VelocityFeedforwardGain | Specify the velocity feedforward gain (any positive number). |
| AccelerationFeedforwardGain | Specify the acceleration feedforward gain (any positive number). |
| PositionProportionalGain | Specify the position proportional gain (any positive number). |
| PositionIntegralGain | Specify the position integral gain (any positive number). |
| VelocityProportionalGain | Specify the velocity proportional gain (any positive number). |
| VelocityIntegralGain | Specify the velocity integral gain (any positive number). |
| VelocityScaling | Specify the velocity scaling attribute that is used to convert the output of the servo loop into equivalent voltage to an external velocity servo drive. |
| TorqueScaling | Specify the torque scaling attribute that is used to convert the acceleration of the servo loop into equivalent % rated torque to the motor. |
| OutputLPFilterBandwidth | Specify the bandwidth in Hertz of the servo low-pass digital output filter. |
| IntegratorHoldEnable | Type **Disabled** or **Enabled**. |
| PositionDifferentialGain | Specify a position differential gain (PosD) to help predict a large overshoot ahead of time and attempt to correct before the overshoot actually occurs. |
| DirectionalScalingRatio | Specify the ratio between the extend direction gain and the retract direction gain. |
| MaximumPositiveTravel | Specify the maximum positive position (any positive number) to be used for software overtravel checking in position units. |
| MaximumNegativeTravel | Specify the maximum negative position (any positive number) to be used for software overtravel checking, in position units. |
| PositionErrorTolerance | Specify the how position error the servo module tolerates (any positive number) before it issues a position error fault. |
| PositionLockTolerance | Specify the maximum position error the servo module accepts (any positive number) to indicate that the position lock status bit is set. |
| OutputLimit | Specify the maximum servo output voltage of a physical axis (any positive number). |
| DirectDriveRampRate | Specify the rate at which the analog output changes from the current value to the requested value when an MDO command is given (if ramp control is enabled). The ramp rate is specified in Volts per second. |
| OutputOffset | Specify a fixed voltage value (-10...10V) to add to the servo output value to correct axis drift. |
| VelocityOffset | Specify a dynamic velocity correction to the output of the position servo loop, in position units/second (any positive number). |
| TorqueOffset | Specify a dynamic torque command correction to the output of the velocity servo loop as a percentage of the velocity servo loop output (-100...100). |
| FrictionCompensation | Specify the percentage (0...100) of output level added to a positive current servo output value, or subtracted from a negative current servo output value to move an axis stuck in place from static friction. |

| Attribute | Description |
|---|---|
| FrictionCompensationWindow | This window is defined as:<br>command position - window attribute to command position + window attribute<br>While the command velocity is zero and the actual position is within this window, the friction compensation, or deadband compensation, for hydraulics, is applied proportionally to the position error. While the command velocity is non-zero, the full friction compensation is applied. |
| BacklashStabilizationWindow | The window controls the backlash stabilization feature in the servo control loop. Mechanical backlash is a common problem in applications that use mechanical gearboxes. |
| BacklashReversalOffset | Specify the backlash reversal error to compensate for positional inaccuracy introduced by mechanical backlash. |
| HardOvertravelFaultAction | Specify the fault action taken when a hardware overtravel error occurs. Type **Shutdown**, **Disable Drive**, **Stop Motion**, or **Status Only**. |
| SoftOvertravelFaultAction | Specify the fault action taken when a software overtravel error occurs. Type **Shutdown**, **Disable Drive**, **Stop Motion**, or **Status Only**. |
| PositionErrorFaultAction | Type **Shutdown**, **Disable Drive**, **Stop Motion**, or **Status Only**. |
| FeedbackFaultAction | Specify the fault action to be taken when a feedback loss condition is detected. Type **Shutdown**, **Disable Drive**, **Stop Motion**, or **Status Only**. |
| FeedbackNoiseFaultAction | Specify the fault action to be taken when excessive feedback noise is detected. Type **Shutdown**, **Disable Drive**, **Stop Motion**, or **Status Only**. |
| DriveFaultAction | Specify the fault action to be taken when a drive fault condition is detected. Type **Shutdown**, **Disable Drive**, **Stop Motion**, or **Status Only**. |
| TestIncrement | Specify the amount of distance traversed by the axis when executing the output and feedback test (any positive number). |
| TuningTravelLimit | Specify the tuning travel limit in revolutions (any positive number). |
| TuningSpeed | Specify the tuning speed in revolutions per second (any positive number). |
| TuningTorque | Specify the tuning torque % rated (0...300). |
| DampingFactor | Specify the damping factor (0.5...2). |
| DriveModelTimeConstant | Specify the drive model time constant (1.0e-6f...1). |
| PositionServoBandwidth | Specify the maximum allowable value for position bandwidth (0.001F...1000), given the damping factor. This parameter is disabled if the loop configuration is set to velocity. |
| VelocityServoBandwidth | Specify the unity gain bandwidth that is to be used to calculate the subsequent gains for a motion apply axis tuning (MAAT) instruction (0.001F...1000). |
| TuningConfigurationBits | Specify the tuning configuration bits. Type a hexadecimal number. |
| TorqueLimitSource | Type **Not Limited**, **Negative Limit**, **Positive Limit**, **Bridge Limit**, **I(t) Limit**, or **Motor Limit**. |
| DriveUnit | Specify the units of the drive. Type **us/in** or **m/s**. |
| PositionDataScaling | Specify the scaling method used on position values (0...255). |
| PositionDataScalingFactor | Specify the scaling factor for all position data in a drive (1...65535). |
| PositionDataScalingExp | Specify the scaling exponent for all position data in a drive (-32768...32767). |
| VelocityDataScaling | Specify the scaling method to use for all velocity values (0...127). |
| VelocityDataScalingFactor | Specify the scaling factor for all velocity data (1...65535). |
| VelocityDataScalingExp | Specify the scaling exponent for all velocity data (-32768...32767). |
| AccelerationDataScaling | Specify the scaling method for all acceleration values (0...127). |
| AccelerationData ScalingFactor | Specify the scaling factor for all acceleration data (1...65535). |
| AccelerationDataScalingExp | Specify the scaling exponent for all acceleration data (-32768...32767). |
| TorqueDataScaling | Specify the scaling method for all torque values (0...127). |

| Attribute | Description |
|---|---|
| TorqueDataScalingFactor | Specify the scaling factor for all torque values (1...65535). |
| TorqueDataScalingExp | Specify the scaling exponent for all torque values (-32768...32767). |
| DrivePolarity | Specify the polarity of the servo loop of the drive. Type **Custom**, **Positive**, or **Negative**. |
| MotorFeedbackType | Specify the type of motor associated with the selected motor (MotorCatalogNumber). If you specify **NONE** for the motor, you must specify a feedback type. |
| MotorFeedbackResolution | Specify the resolution of the motor (1...2147483647). |
| AuxFeedbackType | Specify the type of auxiliary feedback device. |
| AuxFeedbackResolution | Specify the resolution of the auxiliary feedback device (1...2147483647). |
| MotorFeedbackUnit | Specify the units for motor feedback. Type **Rev**, **Inch**, or **Millimeter**. |
| AuxFeedbackUnit | Specify the units for auxiliary feedback. Type **Rev**, **Inch**, or **Millimeter**. |
| OutputNotchFilterFrequency | Specify the frequency of the digital notch filer of the drive (0...10,000.0). |
| VelocityDroop | Specify the velocity droop (any positive number). |
| VelocityLimitBipolar | Specify the velocity limit symmetrically in both directions (any positive number). |
| AccelerationLimitBipolar | Specify the acceleration and deceleration limits for the drive (any positive number). |
| TorqueLimitBipolar | Specify the torque limit symmetrically in both directions (0...1000.0). |
| VelocityLimitPositive | Specify the maximum allowable velocity in the positive direction (any positive number). |
| VelocityLimitNegative | Specify the maximum allowable velocity in the negative direction (any positive number). |
| VelocityThreshold | Specify the velocity threshold limit (any positive number). |
| VelocityWindow | Specify the limits of the velocity window (any positive number). |
| VelocityStandstillWindow | Specify the velocity limit for the standstill window (any positive number). |
| AccelerationLimitPositive | Specify the maximum acceleration ability of the drive (any positive number). |
| AccelerationLimitNegative | Specify the maximum acceleration ability of the drive (any negative number). |
| TorqueLimitPositive | Specify the maximum torque in the positive direction (0...1000.0). |
| TorqueLimitNegative | Specify the maximum torque in the negative direction (-1000.0...0). |
| TorqueThreshold | Specify the torque threshold (0...1000.0). |
| DriveThermalFaultAction | Specify the fault action to be taken when a drive thermal fault is detected. Type **Shutdown**, **Disable Drive**, **Stop Motion**, or **Status Only**. |
| MotorThermalFaultAction | Specify the fault action to be taken when a motor thermal fault is detected. Type **Shutdown**, **Disable Drive**, **Stop Motion**, or **Status Only**. |
| DriveEnableInputFaultAction | Specify the fault action to be taken when a drive enable input fault is detected. Type **Shutdown**, **Disable Drive**, **Stop Motion**, or **Status Only**. |
| StoppingTorque | Specify the amount of torque available to stop the motor (0...1000). |
| StoppingTimeLimit | Specify the maximum amount of time that the drive amplifier will remain enabled while trying to stop (0...6553.5). |
| BrakeEngageDelayTime | Specify the amount of time that the drive maintains torque when the servo axis is disabled, and the drive decelerates to a minimum speed (0...6.5535). |
| BrakeReleaseDelayTime | Specify amount of time that the drive ignores command values from the controller when the servo axis is enabled, and the drive activates the torque (0...6.5535). |
| PowerSupplyID | Specify the power supply ID (any positive number). |
| BusRegulatorID | Specify the bus regulator ID (any positive number). |
| PWMFrequencySelect | Specify High Frequency or Low Frequency. |
| LoadInertiaRatio | Specify the load inertia ratio (any positive number). |
| AmplifierCatalogNumber | Specify the catalog number of the amplifier to which this axis is connected. |

| Attribute | Description |
|---|---|
| AuxFeedbackRatio | Specify the auxiliary feedback ratio (any positive number). |
| ContinuousTorqueLimit | Specify the maximum torque limit (0...200). |
| ResistiveBrakeContactDelay | Specify amount of time to delay resistive brake contact. |
| ConfigurationProfile | Specify the minimum set of attributes the drive can support.<br>**Specify:** **To:**<br>0 Rockwell classic (identifies past systems for backward compatibility)<br>1 Packaging (identifies packaging applications) |
| RegistrationInputs | Specify the number of drive-resident (probe) inputs. Up to two registration inputs per axis. |
| MaximumAccelerationJerk | Specify the value motion instructions used to determine the maximum acceleration jerk rate to apply to the axis when acceleration jerk is specified as a percent of the maximum. This value is only used by a S-curve profile. |
| MaximumDecelerationJerk | Specify the value motion instructions used to determine the maximum deceleration jerk rate to apply to the axis when deceleration jerk is specified as a percent of the maximum. This value is only used by a S-curve profile. |
| DynamicsConfigurationBits | Specify the S-curve profile.<br>**Specify:** **To:**<br>0 Reduce S-curve stop delay<br>1 Prevent S-curve velocity reversals |
| PhaseLossFaultAction | Specify how the axis responds to a drive fault. The default is 1 (disable drive).<br>**Specify:** **To:**<br>0 Shutdown<br>1 Disable drive<br>2 Stop command<br>3 Status only |
| HomeTorqueLevel | Specify the torque limit when using one of the torque homing modes. Type the percent (0...TorqueLimitPositive) of continuous torque. The default is 0%. |
| InputPowerPhase | Specify the power phase operation of a Kinetix 2000 drive. Type **0** for three-phase power; type **1** for single-phase power. |
| MotorRatedContinuousCurrent | The nameplate AC continuous current rating of the motor (any positive number). This is a database number and should not be changed. |
| MotorRatedPeakCurrent | The peak or intermittent current rating of the motor (any positive number). This is a database number and should not be changed. |
| RotaryMotorInertia | The unloaded inertia of a rotary motor. |
| RotaryMotorRatedSpeed | The nameplate rated speed of a rotary motor (any positive number). This is a database number and should not be changed. |
| LinearMotorRatedSpeed | The nameplate rated speed of a linear motor (any positive number). This is a database number and should not be changed. |
| LinearMotorMass | The unloaded moving mass of a linear motor (any positive number). This is a database number and should not be changed. |
| MotorData | The motor data associated with the currently selected catalog number. This should not be changed. |
| AdditionalBusCapacitance | Specify the Additional Bus Capacitance.<br><br>Valid values are in the range of 0...65535 |
| InterpolatedPositionConfiguration | Specify the Interpolated Position Configuration.<br><br>Type a hexadecimal number. |
| AxisUpdateSchedule | Specify an enumeration that indicates which update rate, the Base, Alternate 1 or Alternate 2 update period, to use to update the Axis. |

| Attribute | Description |
|---|---|
| ConverterMotoringPowerLimit | Limits the amount of motoring power allowed to transfer from the AC Line to the motor via the DC Bus. Converter Rated is defined as the Converter Rated Input Power attribute value. |
| ConverterRegenerativePowerLimit | Limits the amount of regenerative power allowed to transfer from the DC Bus to the converter. Since this is regenerative power, the value of the limit is negative. Converter Rated is defined as the Converter Rated Input Power attribute value. |
| ConverterOvertemperatureUserLimit | Sets the user limit for the Converter Overtemperature UL exception. |
| ConverterThermalOverloadUserLimit | Sets the user limit for the Converter Thermal Overload UL exception. |
| ConverterGroundCurrentUserLimit | Sets the user limit for the Converter Ground Current UL exception. |
| CIPAxisExceptionAction2 | A 64-element array of enumerated bytes that specifies the action for the associated extended standard exception. |
| CIPAxisExceptionAction2-RA | A 64-element array of enumerated bytes that specifies the action for the associated extended Rockwell Automation specific axis exception. |
| SafetyFaultAction | An enumerated value that specifies the action taken in the event of a safety fault condition reported by the Safety Core. |
| BusObserverConfiguration | Enumerated value that configures operation of the Bus Observer. The Bus Observer dynamically measures the active current applied to the DC Bus for the purpose of bus impedance compensation. Selecting the **Voltage Estimate** configures the observer to dynamically estimate voltage based on an internal model of the DC Bus. When **Voltage Estimate** is selected, this signal is applied to the voltage loop to provide superior control loop performance. The Voltage Estimate may be used in combination with the Bus Observer by selecting **Bus Observer with Voltage Estimate**. |
| BusObserverBandwidth | Determines the proportional gain, Kbop, of the Bus Observer. This value represents the unity gain bandwidth of the Bus Observer. |
| BusObserverIntegratorBandwidth | Determines the Bus Observer integral gain, Kboi that together with the Kbop, multiplies the integrated error signal within the observer. This value represents the bandwidth of the integrator beyond which the integrator is ineffective. A value of 0 for this attribute disables the integrator. |
| ConverterACInputPhasing | Selects whether the Converter input power to AC line is Single-Phase or Three-Phase. |
| ConverterACInputVoltage | Configures the converter for the intended AC line voltage during normal operation. |
| ConverterPre-ChargeOverloadUserLimit | Sets the user limit for the Converter Pre-Charge Overload UL exception. |
| TotalDCBusCapacitance | Represents the combined capacitance of the regenerative converter and external DC Bus capacitance. |
| ExternalDCBusCapacitance | Represents the combined capacitance of all the external devices that share the DC Bus output of the regenerative converter. |
| ConverterModelTimeConstantBase | Floating point value that represents the lumped model time constant associated with the regenerative converter device for computing loop gains. This attribute stores the original Converter Model Time Constant value for subsequent upload. The Converter Model Time Constant Base is computed based on the converter current loop bandwidth, the bus voltage loop update time and the bus voltage feedback sample period according to the following formula:<br>CMTC_Base = 2 * 1/(2*PI*Current Loop Bandwidth(Hz)) + Bus Voltage Loop Update Period |
| ConverterCurrentLoopBandwidthBase | Floating point value that represents the default bandwidth for the active and reactive current loops for the regenerative converter. This attribute stores the original default Converter Current Loop Bandwidth value that was used to compute the Converter Model Time Constant that is the basis for tuning the converter. |
| ConverterRatedCurrent | Floating point value that represents the continuous output current rating associated with the regenerative converter and used to compute the System Capacitance scaling attribute value from the Total Capacitance of the DC Bus. This attribute stores the original Converter Rated Current value for subsequent upload. |
| ConverterRatedPeakCurrent | Floating point value that represents the peak output current rating associated with the regenerative converter and used together with the Converter Rated Current to compute the default Converter Current Vector Limit attribute value. This attribute stores the original Converter Rated Peak Current value for subsequent upload. |
| ConverterRatedVoltage | Floating point value that represents the input voltage rating associated with the regenerative converter and used to compute the Bus Voltage Set Point attribute value. This attribute stores the original Converter Rated Voltage value for subsequent upload. |
| ConverterDCBusCapacitance | Floating point value that represents the internal bus capacitance of the regenerative converter and is used to compute the System Capacitance scaling attribute. This attribute stores the original Converter DC Bus Capacitance value for subsequent upload. |

<table>
<tr><th>Attribute</th><th>Description</th></tr>
<tr><td>ConverterConfiguration</td><td>Attribute that determines the general control behavior of the regenerative or low harmonic AC/DC converter axis instance.<br>This attribute sets the Converter Control Mode attributes according to the following table:<table><tr><th>Converter Config</th><th>Converter Control Mode</th></tr><tr><td>Bus Voltage Control</td><td>Bus Voltage Control</td></tr><tr><td>Active Current Control</td><td>Active Current Control</td></tr></table></td></tr>
<tr><td>ConverterControlMode</td><td>Determines the basic mode of operation for the Regenerative Converter.<br>When Bus Voltage Control is selected, the converter controls the DC bus voltage output of the converter. The output of the DC bus control loop drives an inner Active AC Line current control loop to maintain the commanded DC bus voltage level established by the Bus Voltage Set Point.<br>When Active Current Control is selected, the converter disables DC bus voltage regulation and directly controls the Active AC Line current component based on the Active Current Command.<br>This value is derived by the controller from the Converter Configuration attribute value during initial configuration.</td></tr>
<tr><td>ReactivePowerControl</td><td>When Reactive Power Control is enabled, the regenerative converter works solely as a reactive power compensation device by injecting reactive power to the grid. This is typically done to improve the power factor on the plant floor or to stabilize AC line voltage. In this mode, the converter does not transfer active power to associated drives on the DC Bus. Instead, all of the converter's rating capacity is dedicated to reactive power correction to the grid. When enabled, the Reactive Power Control function is effective regardless of the configured Converter Control Mode.</td></tr>
<tr><td>ConverterStartupMethod</td><td>Specifies the method to be used to initiate transition of the regenerative converter axis from the Stopped state to the Starting state.</td></tr>
<tr><td>ACLineVoltageTimeConstant</td><td>Sets the low pass filter time constant applied to the AC Line Voltage to determine the AC Line Voltage Nominal attribute value.</td></tr>
<tr><td>ConverterACInputFrequency</td><td>Selects the nominal frequency of the AC Line connected to the converter.</td></tr>
<tr><td>ACLineVoltageUnbalanceLimit</td><td>Sets the maximum allowed voltage unbalance between the AC line phases. Exceeding this limit results in a Converter AC Unbalance exception.<br>Nominal voltage is defined by the AC Line Voltage Nominal attribute.</td></tr>
<tr><td>ACLineCurrentUnbalanceLimit</td><td>Sets the maximum allowed current unbalance between the AC line phases. Exceeding this limit results in a Converter AC Unbalance exception.<br>Rated current is defined by the Converter Rated Input Current attribute.</td></tr>
<tr><td>ACLineSyncErrorTolerance</td><td>Sets the maximum allowed phase error associated with the AC line synchronization function of the regenerative converter. Exceeding this limit results in an AC Line Sync Loss exception.</td></tr>
<tr><td>ACLineSourceSelect</td><td>Selects which AC Line source is active for the converter and applies the configured impedance and power rating of that source to the converter's control structure.</td></tr>
<tr><td>ACLineSourceImpedance</td><td>Impedance of the AC line source as a percent of the transformer or generator impedance rating.</td></tr>
<tr><td>ACLineSourcePower</td><td>Sets the power rating of the transformer or generator feeding power to the converter as a percent of the converter's power rating.</td></tr>
<tr><td>ACLineSourceImpedance-Alternate</td><td>Impedance of the alternate AC line source as a percent of the transformer or generator impedance rating.</td></tr>
<tr><td>ACLineSourcePower-Alternate</td><td>Selects the power rating of the alternate transformer or generator feeding power to the converter as a percent of the converter's power rating.</td></tr>
<tr><td>BusVoltageSetPoint</td><td>Sets the reference voltage used to actively regulate the DC Bus Voltage of the converter when in the Running state and the Bus Voltage Reference Source is set to Manual.</td></tr>
<tr><td>BusVoltageReferenceSource</td><td>Selects between Automatic and Manual source for the Bus Voltage Reference. Automatic (default) selection allows converter to optimize the Bus Voltage Reference for best converter performance. With the Manual selection, the converter uses the user-configured Bus Voltage Set Point value for the Bus Voltage Reference signal.</td></tr>
<tr><td>BusVoltageLoopBandwidth</td><td>Determines the proportional gain, Kbp, of the bus voltage loop that multiplies the Bus Voltage Error signal. This value represents the unity gain bandwidth of the bus voltage loop.</td></tr>
</table>

| Attribute | Description |
|---|---|
| BusVoltageIntegratorBandwidth | Determines the bus voltage loop integral gain, Kbi, which together with the Kbp, multiplies the integrated Bus Voltage Error signal. This value represents the bandwidth of the bus voltage integrator beyond which the integrator is ineffective. A value of 0 for this attribute disables the integrator. |
| BusVoltageRateLimit | Sets the DC Bus rate limit for the Bus Voltage Set Point that becomes the DC Bus Reference signal when the Bus Voltage Reference Source is set to **Manual**. |
| BusVoltageErrorTolerance | Determines the absolute maximum Bus Voltage Error value that can be tolerated without causing an Excessive Bus Voltage Error exception. |
| BusVoltageErrorToleranceTime | Determines the maximum amount of time that the Bus Voltage Error Tolerance can be exceeded without generating an exception. |
| ReactivePowerSetPoint | Sets the reference current used to actively regulate the AC Line Reactive Power of the converter when in the Running state. Attribute units are expressed in percent for Converter Rated Output Power.<br>Positive value indicates lagging kVAR and negative value indicates leading kVAR. |
| ReactivePowerRateLimit | Sets the active current rate limit for AC Line Reactive Power Set Point input. The output of the Reactive Power Rate Limit function is the AC Line Reactive Power Reference signal. Attribute units are expressed in percent for Converter Rated Output Power per second. |
| SystemCapacitance | Scaling gain value that converts voltage rate commanded by the bus voltage control loop into equivalent active current, expressed as a percent of the converter's current rating. Properly set, this value represents the total system capacitance of the DC bus. |
| ActiveCurrentCommand | Sets the reference current used to actively regulate the Active Current of the converter when in the Running state and configured for AC Line Current Control mode.<br>Positive value implies motoring current, while a negative value implies regenerative current. |
| ReactiveCurrentCommand | Sets the reference current used to actively regulate the Reactive Current of the converter when in the Running state and configured for AC Line Current Control mode.<br>Positive value implies reactive current (lagging relative to voltage) is consumed by the converter, while a negative value implies reactive current (leading relative to voltage) is produced by the converter. |
| ActiveCurrentTrim | Additional current command added to the active current reference summing junction. |
| ActiveCurrentLowPassFilterBandwidth | Break frequency for the low pass filter applied to active current reference signal. A value of 0 for this attribute disables this feature. |
| ActiveCurrentNotchFilterFrequency | Center frequency of the notch filter applied to the active current reference signal. A value of 0 for this attribute disables this feature. |
| ActiveCurrentRateLimit | Sets the magnitude limit on the rate of change of the converter's active current reference signal. This attribute only applies when configured for AC Line Current Control mode. |
| ReactiveCurrentRateLimit | Sets the magnitude limit on the rate of change of the converter's reactive current reference signal. This attribute only applies when configured for AC Line Current Control mode. |
| ConverterCurrentLimitSource | Represents the operative source of a converter current limit when a current limit condition occurs. |
| ConverterCurrentLoopBandwidth | AC Line Current Loop Proportional Gain value that multiplies the active and reactive AC Line Current Error signals. This value directly determines the bandwidth of the active and reactive AC line current loops. |
| ConverterCurrentIntegratorBandwidth | AC Line Current Loop Integral Gain value that, together with Kcp, multiplies the active and reactive AC Line Current Error signals before applying them to the active and reactive AC Line Current Integrator Error accumulators. This value represents the bandwidth of the velocity integrator beyond which the integrator is ineffective. A value of 0 for this attribute disables the integrators. |
| ConverterCurrentVectorLimit | Sets the value applied to current vector limiter to provide a configurable limit to the magnitude of the current vector of the converter's active and reactive current reference signals. |
| ACLineVoltageSagAction | Sets the reaction to a Voltage Sag condition when any one of the AC Line phase voltages drops below a hard-coded threshold in the device or the configured AC Line Voltage Sag Threshold. This provides a specific (configured) response to an incoming AC Line Voltage Sag condition while the device is running. |
| ACLineVoltageSagThreshold | Sets the level for AC Line Voltage Sag as percent of nominal AC Line Voltage. Nominal voltage is defined by the AC Line Voltage Nominal attribute. Measured AC Line Voltage values less than this threshold indicate an AC Line Voltage Sag condition. |

| Attribute | Description |
|---|---|
| ConverterInputPhaseLossAction | Sets the reaction to an AC input phase loss condition. This provides a specific (configured) response to an incoming phase loss while the converter is running. |
| ConverterInputPhaseLossTime | When the Converter Input Phase Loss Action is set to Ride Thru, this attribute sets the timeout value before a Converter AC Single Phase Loss Exception is generated by the device in response to the Converter Input Phase Loss condition. A value of 0 in this case results in an immediate exception. |
| ACLineFrequencyChangeAction | Sets the converter's reaction when the rate of change of the AC line frequency exceeds a hard-coded threshold or the configured Frequency Change Threshold. |
| ACLineFrequencyChangeThreshold | Sets the level of AC line frequency change that results in the AC Line Frequency Change condition. |
| ACLineFrequencyChangeTime | When the AC Line Frequency Change Action is set to Ride Thru, this attribute sets the timeout value before an AC Line Frequency Change Exception is generated by the converter in response to an AC Line Frequency Change condition. A value of 0 in this case results in an immediate exception. |
| ACLineSyncLossAction | Sets the reaction to a loss of AC line synchronization by the converter's line synchronization function (for example, PLL). This provides a specific (configured) response to an incoming line synchronization loss condition while the converter is running. |
| ACLineSyncLossTime | When the AC Line Sync Loss Action is set to **Ride Thru,** this attribute sets the timeout value before an AC Line Sync Exception is generated by the converter in response to an AC Line Sync Loss condition. A value of 0 in this case results in an immediate exception. |
| ConverterOverloadAction | Selects the device's response to a converter overload condition based on an I2T or converter thermal model based overload protection method. When a converter thermal model is employed, the converter overload condition occurs when the converter thermal model indicates that the Converter Capacity has exceeded the Converter Overload Limit. In the case of the I2T overload protection method, the converter overload condition occurs when the converter current, in percent of rated continuous converter current, exceeds the Converter Overload Limit.<br>The Converter Overload Action provides opportunities to mitigate the overload condition without stopping operation. Converter Overload Action functionality is independent of the converter overload exception action functionality.<br>An overload alarm condition can also be generated by exceeding the limits of the device's power block thermal model that includes switching losses that have a dependency on the PWM Frequency.<br>No explicit action is taken by the device in the overload condition if None is the selected overload action. Selecting the Current Foldback action, however, results in a reduction of the converter current in proportion to the percentage difference between Converter Capacity and the Converter Overload Limit, or in the case of the I2T overload protection method, in proportion to the difference between the converter current, in percent of rated continuous converter current, and the Converter Overload Limit.<br>In addition to the configured Converter Overload Action being triggered by a Converter Overload condition, the configured Converter Overload Action can also be triggered by Converter L1, L2, or L3 Overload conditions. |
| ACLineOvervoltageUserLimit | Sets the high voltage limit as a percent of Converter Rated Input Voltage for the AC line source. |
| ACLineUndervoltageUserLimit | Sets the low voltage limit as a percent of Converter Rated Input Voltage for the AC line source. |
| ACLineOvervoltageUserLimit-Alternate | Sets the high voltage limit as a percent of Converter Rated Input Voltage for the alternate AC line source. |
| ACLineUndervoltageUserLimit-Alternate | Sets the low voltage limit as a percent of Converter Rated Input Voltage for the alternate AC line source. |
| ACLineHighFreqUserLimit | Sets the high frequency limit as the difference from the nominal AC line frequency. |
| ACLineLowFreqUserLimit | Sets the low frequency limit as the difference from the nominal AC line frequency. |
| ACLineHighFreqUserLimit-Alternate | Sets the high frequency limit as the difference from the nominal alternate AC line source frequency. |
| ACLineLowFreqUserLimit-Alternate | Sets the low frequency limit as the difference from the nominal alternate AC line source frequency. |
| ConverterHeatsinkOvertemperatureUserLimit | Sets a User Limit for the regenerative converter power structure heatsink temperature. Exceeding this value generates a Converter Overtemperature UL exception. |
| ACLineOverloadUserLimit | Sets a User Limit for thermal overload of Line 1, 2, and 3 components as a percent of their rated thermal capacity. Exceeding this value generates a Converter Thermal Overload UL exception. |

| Attribute | Description |
|---|---|
| ConverterCurrentLoopTuningMethod | Method used to configure the responsiveness of the active and reactive current loops.<br>With the **Direct** method, the current loop response is determined directly by the Converter Current Integrator Bandwidth value that maps to the integral gain, Kci, of the converter's current loop. When configured for the Direct tuning method, the Converter Current Loop Damping attribute value has no effect on the current control loops.<br>With the **Calculated** method, the loop response is determined by the Converter Current Loop Damping value. The converter uses this value to calculate the appropriate internal current loop integral gain, Kci, based on the Converter Current Loop Bandwidth and known load characteristics of the AC Line. When configured for the Calculated tuning method, the Converter Current Loop Integrator Bandwidth attribute value has no effect on the current control loops. |
| ConverterCurrentLoopDamping | Damping factor that, together with the specified Converter Current Loop Bandwidth value, determines responsiveness of the active and reactive AC line current loops. This attribute may be used as an alternative to directly setting the Kci gain value for the current loops. A damping factor of 1 results in a critically damped current loop. |
| ACLineContactorInputChecking | Controls whether or not the AC/DC converter function checks for the presence and proper operation of the AC Line Contactor using the AC Line Contactor OK input. If AC Line Contactor Checking is Enabled, the presence of AC line voltage at the device when the AC Line Contactor OK input is Inactive shall cause the converter to generate an AC Line Contactor exception. If AC Line Contactor Checking is Disabled then the device shall not check the AC Line Contactor OK input. |
| ConverterModelTimeConstant | The value for the Converter Model Time Constant represents the lumped model time constant for the converter's current loop and is used to calculate Bus Voltage loop bandwidth values. The Converter Model Time Constant is the sum of the converter's turrent loop time constant, and the calculation delay. This value is set by the application based on th specific converter selection. |
| ConverterRatedPower | Floating point value that represents the power rating of the converter. Use this attribute to estimate the default AC Line Source Power value. |
| CurrentLoopBandwidthScalingFactor | Floating point value that represents the scaling factor, based on motor type, which is used to set the factory default value for Torque Loop Bandwidth. This attribute stores the original Current Loop Bandwidth Scaling Factor value for subsequent upload. |
| CurrentLoopBandwidth | Floating point value that represents the bandwidth of the current loop that is used to set the factory default value for Torque Loop Bandwidth. This attribute stores the Current Loop Bandwidth value for subsequent upload. |
| DriveRatedVoltage | Floating point value that represents the RMS voltage rating of the drive that is used to set the factory default value for the Break Voltage associated with V/Hz drives. This attribute stores the original Drive Rated Voltage value for subsequent upload. |
| MaxOutputFrequency | Floating point value that represents the maximum frequency rating of the drive that is used to set the factory default values for Velocity Limits. This attribute stores the original Max Output Frequency value for subsequent upload. |
| ACLineResonanceUserLimit | Sets the User Limit for the AC Line Resonance UL exception based on a percentage of the Converter Output Rated Current.<br>The AC Line Resonance UL exception occurs when the current flowing through the AC line filter in the resonant frequency band, exceeds the user limit for a vendor specified period. |

# COORDINATE_SYSTEM tag

## L5X CoordinateSystem structure

In an.L5X file, coordinate system attributes are in the Data element.

```
<Data Format="CoordinateSystem">
      <CoordinateSystemParameters
            [Coordinate_System_Attributes]/>
</Data>
```

## L5K COORDINATE_SYSTEM structure

In an.L5K file, coordinate system attributes are in the tag statement.

```
<tag_name> : <type>
      [(Coordinate_System_Attributes)]
```

## Coordinate system tag attributes

| Attribute | Description |
|---|---|
| MotionGroupInstance | Type the name of the associated motion group or type **NA**. |
| SystemType | Specify the coordinate system type. Type **Cartesian**, **Articulated Dependent**, **Articulated Independent**, **Delta**, **SCARA Delta**, or **SCARA Independent**. |
| Dimension | Specify the number of axes that this coordinated system supports. Type **1**, **2**, or **3**. |
| Axes | Specify the name of the axes in this coordinated system. |
| CoordinationMode | Specify coordination mode. Type **Primary** or **Ancillary**. |
| CoordinationUnits | Specify units to be used for measuring and calculating motion related values such as position, velocity. Type units that are relevant to your application. |
| ConversionRatioNumerator | The conversion ratio defines the relationship of axis position units to coordination units for each axis. Type the numerator as a float or an integer. |
| ConversionRatioDenominator | The conversion ratio defines the relationship of axis position units to coordination units for each axis. Type the denominator as an integer. |
| CoordinateSystemAutoTagUpdate | Specify whether or not the actual position values of the current coordinated system are automatically updated during operation. To enable auto tag update, type **1**. Otherwise, type **0**. |
| MaximumSpeed | Specify the maximum speed to be used by the coordinated motion instructions in calculating vector speed when speed is expressed as a percent of maximum. |
| MaximumAcceleration | Specify the value for maximum acceleration to be used by the coordinated motion instructions to determine the acceleration rate to apply to the coordinate system vector when acceleration is expressed as a percent of maximum. |
| MaximumDeceleration | Specify the value for maximum deceleration to be used by the coordinated motion instructions to determine the deceleration rate to apply to the coordinate system vector when deceleration is expressed as a percent of maximum. |
| ActualPositionTolerance | Specify the value in coordination units, for actual position to be used by coordinated motion instructions when they have a termination type of actual tolerance. |
| CommandPositionTolerance | Specify the value in coordination units, for command position to be used by coordinated motion instructions when they have a termination type of command tolerance. |
| TransformDimension | Specify the transform dimension. |
| JointRatioNumerator | Specify numerator of the joint ratio. |
| JointRatioDenominator | Specify denominator of the joint ratio. |
| LinkLength1 | Specify the length of Robotic Arm 1. |
| LinkLength2 | Specify the length of Robotic Arm 2. |
| ZeroAngleOffset1 | Specify the rotational angular offset of joint axes 1 in degrees. This is used to shift the 0 degree position of the joint. |
| ZeroAngleOffset2 | Specify the rotational angular offset of joint axes 2 in degrees. This is used to shift the 0 degree position of the joint. |
| ZeroAngleOffset3 | Specify the rotational angular offset of joint axes 3 in degrees. This is used to shift the 0 degree position of the joint. |
| BaseOffset1 | Specify the difference for the first axis between the origin of the robot at the first joint of the robotic arm, as determined by the application's Kinematics internal equations, and the origin defined by the robot manufacturer. |
| BaseOffset2 | Specify the difference for the second axis between the origin of the robot at the first joint of the robotic arm, as determined by the application's Kinematics internal equations, and the origin defined by the robot manufacturer. |
| BaseOffset3 | Specify the difference for the third axis between the origin of the robot at the first joint of the robotic arm, as determined by the application's Kinematics internal equations, and the origin defined by the robot manufacturer. |
| EndEffectorOffset1 | Specify the end effector offset value, which is the distance between the end of the robot arm L2 and the end of the end effector in the x1 dimension. |
| EndEffectorOffset2 | Specify the end effector offset value, which is the distance between the end of the robot arm L2 and the end of the end effector in the x2 dimension. |

| Attribute | Description |
|---|---|
| EndEffectorOffset3 | Specify the end effector offset value, which is the distance between the end of the robot arm L2 and the end of the end effector in the x3 dimension. |
| MaximumAccelerationJerk | Specify the value for maximum acceleration jerk to be used by the coordinated motion instructions to determine the acceleration jerk rate to apply to the coordinate system vector when acceleration jerk is expressed as a percent of maximum. |
| MaximumDecelerationJerk | Specify the value for maximum deceleration jerk to be used by the coordinated motion instructions to determine the deceleration jerk rate to apply to the coordinate system vector when deceleration jerk is expressed as a percent of maximum |
| MasterInputConfigurationBits | Specify the master input configuration bits.<br><br>Type a hexadecimal number. |
| MasterPositionFilterBandwidth | Specify the Master Position Filter Bandwidth. |

# MOTION_GROUP Tag

## L5X MotionGroup structure

In an.L5X file, motion group attributes are in the Data element.

```
<Data Format="MotionGroup">
      <MotionGroupParameters [Motion_Group_Attributes]/>
</Data>
```

## L5K MOTION_GROUP structure

In an.L5K file, motion group attributes are in the tag statement.

```
<tag_name> : <type>
      [(Motion_Group_Attributes)];
```

## Motion Group Tag attributes

| Attribute | Description |
|---|---|
| GroupType | Specify the type of motion group. Type **Warning Enabled** or **Warning Disabled**. |
| CourseUpdatePeriod | Specify the coarse update period in milliseconds (500...3200ms). |
| PhaseShift | Specify the phase shift (0...65,535). |
| GeneralFaultType | Specify whether an error generates a major fault or a non-major fault. Type **Major Fault** or **Non Major Fault**. |
| AutoTagUpdate | Type **Disabled** or **Enabled**. |
| Alternate1UpdateMultiplier | Specify the value that is multiplied by the Base Update Period, which was previously called the Coarse Update Period. The result is displayed as the **Alternate 1 Update Period**. |
| Alternate2UpdateMultiplier | Specify the value that is multiplied by the **Base Update Period**, which was previously called the Coarse Update Period. The result is displayed as the **Alternate 2 Update Period**. |

# HMIBC tag

## L5X HMIBC structure

In an.L5X file, HMIBC attributes are in the Data element.

```
<Tag Name="MyHMIBC" TagType="Base" DataType="HMIBC"
ExternalAccess="Read/Write">
      <Data Format="HMIBC">
       <HMIBCParameters EnableIn="false" ProgFB="false"
BitIndex="0" TerminalCount="0"/>
```

```
      </Data>
</Tag>
```

### L5K HMIBC structure

```
TAG
      myHMIBC : HMIBC (EnableIn := "false",
                       ProgFB := "false");
END_TAG
```

### HMIBC attributes

| L5X Item | L5K Item | Description |
|---|---|---|
| EnableIn | EnableIn | Relay Ladder: Corresponds to the rung state. Does not affect processing.<br>Function Block: If cleared, the instruction does not execute and outputs are not updated. If set, the instruction executes. Default is set.<br>Structured Text: No effect. The instruction always executes.<br>Value: **true** or **false** |
| ProgFB | ProgFB | Program Feedback. For use by the user program. Not used by the instruction.<br>Value: **true** or **false** |
| BitIndex | | L5X Only: Export only, ignored on import<br>Value: 0 - 65535 |
| TerminalCount | | L5X Only: Export only, ignored on import<br>Value: 0 - 65535 |

## SAFETY tag

In an.L5X file, safety attributes are in the Data element.

### L5X safety structure

```
<Data Foamt="Safety">
      <SafetyParameters [Safety_Attributes]/>
</Data>
```

### L5K SAFETY structure

In an.L5K file, safety attributes are in the tag statement.

```
<tag_name> : <type>
      [(Safety_Attributes)] := value;
```

### Safety tag attributes

A safety produced or safety consumed tag (Class = Safety) has these attributes, and the attributes for a standard tag.

| Attribute | Description |
|---|---|
| TimeoutMultiplier | Specify the timeout multiplier (default = 2) for a safety controller system. This value determines the RPIs of time to wait for a packet before declaring a time out. This translates into the number of messages that may be lost before declaring a connection error. A Timeout Multiplier of 1 indicates that no messages may be lost, which means there must be a packet every RPI interval. A Timeout Multiplier of 2 indicates that 1 message may be lost, which means as long as a packet is seen in 2 times the RPI, no time-out will occur. Type a number from 1...4, inclusive.<br>This attribute applies only to safety consumed tags.<br><br>For L5X, this attribute is on the <ConsumeInfo> element. |

| Attribute | Description |
|---|---|
| NetworkDelayMultiplier | Specify the network delay multiplier (default = 100%) for a safety controller system. This value lets you reduce or increase the connection reaction time limit in cases where the transport time of the message is significantly less or more than the RPI. This may be the case when the RPI of an output connection is the same as that of a lengthy task period. Type a percentage from 10...300, inclusive.<br>This attribute applies only to safety consumed tags.<br><br>For L5X, this attribute is on the <ConsumeInfo> element. |
| ReactionTimeLimit | Specify the connection reaction time limit for a safety controller system. The Logix Designer application calculates the connection reaction time limit as a function of the RPI, timeout multiplier, and network delay multiplier. The connection reaction time limit is automatically recalculated if any of the above values change.<br>This attribute applies only to safety consumed tags.<br><br>For L5X, this attribute is on the <ConsumeInfo> element. |
| MaxObservedNetworkDelay | L5X only. MaxObservedNetworkDelay is a measure of the longest time the data of a safety connection is delayed from transporting the safety packets over the network. This attribute is exported for informational purposes only and is ignored on import.<br><br>This attribute is on the <ConsumeInfo> element. |
| Unicast | For a safety consumed tag, specify if the EtherNet/IP connection is unicast. Specify **yes** for unicast or **no** to remain multicast. For L5X format, specify **true** or **false**.<br>On export, only appears if the path to the producing module crosses an EtherNet/IP network and the producing module supports unicast. |
| UnicastPermitted | For a safety produced tag, specify whether let the EtherNet/IP connection be unicast. Specify **yes** for unicast or **no** to remain multicast. For L5X format, specify **true** or **false**. |

# Tag initial values

## L5X initial tag value

In an.L5X file, place the initial value of the tag in the Data element. See Chapter 1 Data Formats on page 43 for more information on Data elements.

```
<Data>
      initial_value
</Data>
```

If the tag is a string tag, specify a Format attribute.

```
<Data Format="String" Length="value">
      <[CDATA['string text']]>
</Data>
```

In L5X format, the string is padded with zeros (00) to fill its maximum of 82 characters.

## L5K initial TAG value

In an.L5K file, place the initial value of the tag in the tag statement.

```
<tag_name> : <type> [(Attributes)][:= <initial_value>];
```

If the tag is a string tag, specify a STRING type.

<*tag_name*> : STRING := [<*length*>, '*string_text*$00... $00'];

The string is padded with zeros ($00) to fill its maximum of 82 characters.

## Add-On Instruction tag values

The Logix Designer application optimizes user memory usage by minimizing unused space in tags whose type is an Add-On Instruction data type.

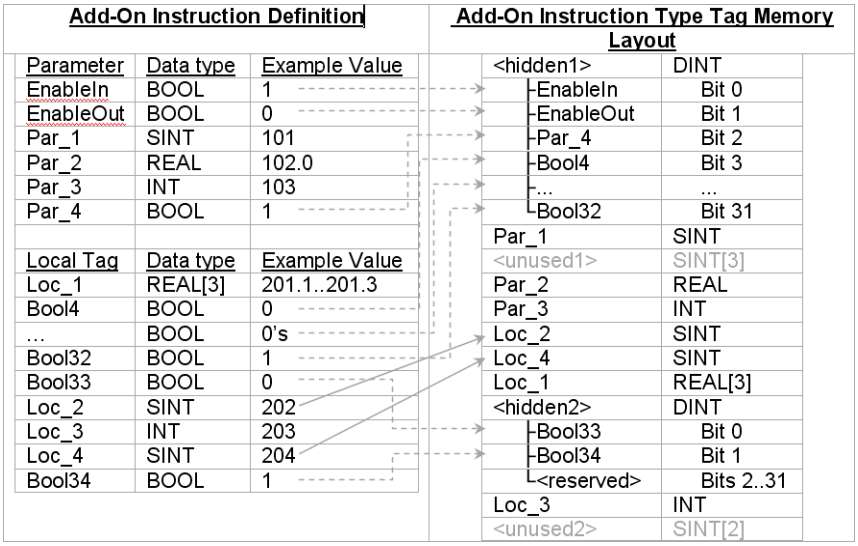
The values for the Input & Output parameters are represented in the order as they are listed in the Add-On Instruction definition. This rule applies to all parameter data types except for BOOL. In order to reduce the data size of a tag of Add-On Instruction type, BOOL parameters and BOOL local tags are collected and packed in one or more hidden DINT members. Up to 32 BOOLs are packed into each hidden DINT member. That is, the first BOOL is at bit 0, the second BOOL is at bit 1. The hidden DINT members are located at the position where the 1st, 33rd, 65th, (n*32+1) BOOL is located. See the example.

The values for the local tags are represented in the order listed in the Add-On Instruction definition except as noted here. BOOL local tags are packed with BOOL parameters. In order to reduce the data size of an Add-On Instruction type tag, SINT and INT tags may be repositioned.

This is done because all types except SINT and INT must be aligned on a four-byte boundary. When a SINT precedes a DINT, there are three unused bytes between the two members. To optimize memory usage, the unused space is filled with other SINTs and/or INTs when possible. Note that the free space optimization begins immediately after the last parameter. See the example.

## Add-On Instruction tag values example

This example demonstrates the tag memory layout of an Add-On Instruction definition. The dashed lines show where the BOOL members are located in memory. The solid lines indicate where a local member is repositioned to minimize unused space.




The L5K representation of an Add-On Instruction type tag containing the example values is: [-2147483643, 101, 102.0, 103, 202, 204, [201.1, 201.2, 201.3], 2, 203].

- The <unused1> memory area aligns the member, Par_2, on a 4-byte boundary. All types except for SINT and INT must be aligned on a 4-byte boundary. INTs must be aligned on a 2-byte boundary.
- The <unused2> memory area pads the Add-On Instruction type tag such that the byte length is a multiple of 4.
- Unused data is not included in the L5K format. Unused data is included in the L5X format.
- The simplest way to guarantee that the order of the parameters and local tags in the L5K Add-On Instruction definition matches the order of data values in the L5K Add-On Instruction type tag is to order the tags in the LOCAL_TAGS section such that all SINTs are first, followed by all INTs, and everything else.

## Tag guidelines

Observe these guidelines when defining a tag.

- Tags must be defined after modules and add-on instructions. If there are no modules or add-on instructions, then the tags must be defined after the data types within the controller body.
- Base tags and aliases can be defined out of order within a tag block.
- You cannot define a second dimension without a first dimension or a third dimension without a second dimension.
- The initial values must comply with the tag type and dimensions.
- Whitespace cannot occur within the initial values in L5K format or within the type/dimension specifier.

# Examples

**L5X tag example**

```
- <Tags>
    <Tag Name="myBool" TagType="Alias" Radix="Decimal" AliasFor="Local:1:I.Data.7" />
  - <Tag Name="myProduced" TagType="Produced" DataType="DINT" Radix="Decimal">
      <ProduceInfo ProduceCount="1" ProgrammaticallySendEventTrigger="false"
        UnicastPermitted="false" />
      <Data>03 00 00 00</Data>
      <ForceData>00 00 00 00 00 00 00 00 00 00 00 00</ForceData>
    - <Data Format="Decorated">
        <DataValue DataType="DINT" Radix="Decimal" Value="3" />
      </Data>
    </Tag>
  - <Tag Name="myTimer" TagType="Base" DataType="TIMER">
      <Data>00 00 00 00 05 00 00 00 02 00 00 00</Data>
    - <Data Format="Decorated">
      - <Structure DataType="TIMER">
          <DataValueMember Name="PRE" DataType="DINT" Radix="Decimal" Value="5" />
          <DataValueMember Name="ACC" DataType="DINT" Radix="Decimal" Value="2" />
          <DataValueMember Name="EN" DataType="BOOL" Value="0" />
          <DataValueMember Name="TT" DataType="BOOL" Value="0" />
          <DataValueMember Name="DN" DataType="BOOL" Value="0" />
        </Structure>
      </Data>
    </Tag>
  </Tags>
```

**L5K TAG examples**

```
TAG
       bits : MySint := [0];
       dest : INT (RADIX := Decimal) := 0;
       overflow OF bits.MyBit0 (RADIX := Binary);
       source : REAL (RADIX := Exponential) := 0.0;
       timer : TIMER[3] := [[0,0,100],[0,10,100],[0,0,50]];
END_TAG
```

This example shows forced tag data.

```
TAG
       dint_a : DINT (RADIX := Decimal) := 0;
       int_a : INT (RADIX := Decimal) := 0;
       tag_a : UDT_A (ProduceCount := 2) := [0,0],
       TagForceData := [0,0,0,0,1,0,-1,-1,1,0,-72,34];
END_TAG
```

#### L5X safety tag example

```
- <Tag Name="SafetyConsumed" TagType="Consumed" DataType="SafetyDint"
    Class="Safety">
    <ConsumeInfo Producer="OtherProducer" RemoteTag="MySafetyTag"
      RemoteInstance="0" RPI="20" TimeoutMultiplier="2" NetworkDelayMultiplier="200"
      ReactionTimeLimit="80" MaxObservedNetworkDelay="0" />
    <Data>00 00 00 00 00 00 00 00</Data>
    <ForceData>00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
      00</ForceData>
  - <Data Format="Decorated">
    - <Structure DataType="SafetyDint">
      - <StructureMember Name="Status" DataType="CONNECTION_STATUS">
          <DataValueMember Name="RunMode" DataType="BOOL" Value="0" />
          <DataValueMember Name="ConnectionFaulted" DataType="BOOL"
            Value="0" />
        </StructureMember>
        <DataValueMember Name="Data" DataType="DINT" Radix="Decimal"
          Value="0" />
      </Structure>
    </Data>
  </Tag>
- <Tag Name="SafetyProduced" TagType="Produced" DataType="SafetyDint"
    Class="Safety">
    <ProduceInfo ProduceCount="1" ProgrammaticallySendEventTrigger="true" />
    <Data>00 00 00 00 00 00 00 00</Data>
    <ForceData>00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
      00</ForceData>
  - <Data Format="Decorated">
    - <Structure DataType="SafetyDint">
      - <StructureMember Name="Status" DataType="CONNECTION_STATUS">
          <DataValueMember Name="RunMode" DataType="BOOL" Value="0" />
          <DataValueMember Name="ConnectionFaulted" DataType="BOOL"
            Value="0" />
        </StructureMember>
        <DataValueMember Name="Data" DataType="DINT" Radix="Decimal"
          Value="0" />
      </Structure>
    </Data>
  </Tag>
```

#### L5K safety TAG examples

This example shows a consumed tag in a safety project.

```
safetyConsumed : mypcType (Class := Safety,
       Producer := PeerSafetyController,
       RemoteTag := productCount,
       RemoteFile := 0,
       RPI := 10,
       IncludeConnectionStatus := Yes,
       TimeoutMultiplier := 2,
       NetworkDelayMultiplier := 100,
       ReactionTimeLimit := 29.952) := [[2],[0,0,0]],
       TagForceData :=
```

```
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
```

This example shows a produced tag in a safety project:

```
safetyProduced : mypcType (Class := Safety,
       ProduceCount := 3,
       ProgrammaticallySendEventTrigger := Yes,
       IncludeConnectionStatus := Yes) := [[0],[0,0,0]],
       TagForceData :=
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
```

#### L5X ALARM_ANALOG and ALARM_DIGITAL tag examples

```
- <Tag Name="AnalogAlarmTag" TagType="Base" DataType="ALARM_ANALOG">
  - <Description>
      <![CDATA[ This is an alarm analog tag.  ]]>
    </Description>
  - <Data Format="Alarm">
      <AlarmAnalogParameters EnableIn="false" InFault="false" HHEnabled="true"
        HEnabled="true" LEnabled="true" LLEnabled="true" AckRequired="true"
        ProgAckAll="false" OperAckAll="false" HHProgAck="false" HHOperAck="false"
        HProgAck="false" HOperAck="false" LProgAck="false" LOperAck="false"
        LLProgAck="false" LLOperAck="false" ROCPosProgAck="false"
        ROCPosOperAck="false" ROCNegProgAck="false" ROCNegOperAck="false"
        ProgSuppress="false" OperSuppress="false" ProgUnsuppress="false"
        OperUnsuppress="false" ProgDisable="false" OperDisable="false" ProgEnable="false"
        OperEnable="false" AlarmCountReset="false" In="0.0" HHLimit="50.0"
        HHSeverity="500" HLimit="40.0" HSeverity="500" LLimit="30.0" LSeverity="500"
        LLLimit="20.0" LLSeverity="500" MinDurationPRE="0" Deadband="0.0"
        ROCPosLimit="0.0" ROCPosSeverity="500" ROCNegLimit="0.0" ROCNegSeverity="500"
        ROCPeriod="0.0" />
    - <AlarmConfig>
      - <Messages>
        - <Message Type="HH">
          - <Text Lang="en-US">
              <![CDATA[ Shut down system  ]]>
            </Text>
          </Message>
        + <Message Type="H">
        + <Message Type="L">
        + <Message Type="LL">
        </Messages>
      </AlarmConfig>
    </Data>
  </Tag>
```

```
- <Tag Name="DigitalAlarmTag" TagType="Base" DataType="ALARM_DIGITAL">
  - <Description>
      <![CDATA[ This is an alarm digital tag ]]>
    </Description>
  - <Data Format="Alarm">
      <AlarmDigitalParameters Severity="500" MinDurationPRE="0" ProgTime="DT#1970-01-
        01-00:00:00.000000Z" EnableIn="false" In="false" InFault="false"
        Condition="true" AckRequired="true" Latched="false" ProgAck="false"
        OperAck="false" ProgReset="false" OperReset="false" ProgSuppress="false"
        OperSuppress="false" ProgUnsuppress="false" OperUnsuppress="false"
        ProgDisable="false" OperDisable="false" ProgEnable="false" OperEnable="false"
        AlarmCountReset="false" UseProgTime="false" />
    - <AlarmConfig>
      - <Messages>
        - <Message Type="AM">
          - <Text Lang="en-US">
              <![CDATA[ Monitor Presssure ]]>
            </Text>
          </Message>
        </Messages>
      </AlarmConfig>
    </Data>
  </Tag>
```

**L5K ALARM_ANALOG and ALARM_DIGITAL tag examples**

```
my_alarm2 : ALARM_ANALOG (ALMSG.HH:en-us := "High high alarm message",
       ALMMSG.POS:en-us := "pos alarm message",       ALMMSG.NEG:en-us := "neg alarm message",
       EnableIn := false,  InFault := false,   HHEnabled := true,  HEnabled := false,
       LEnabled := false,  LLEnabled := false,        AckRequired := true,       ProgAckAll := false,
       OperAckAll := false,       HHProgAck := false,        HHOperAck := false,        HProgAck := false,
       HOperAck := false,  LProgAck := false,  LOperAck := false,  LLProgAck := false,
       LLOperAck := false,        ROCPosProgAck := false,    ROCPosOperAck := false,
       ROCNegProgAck := false,    ROCNegOperAck := false,    ProgSuppress := false,
       OperSuppress := false,     ProgUnsuppress := false,   OperUnsuppress := false,
       ProgDisable := false,      OperDisable := false,      ProgEnable := false,       OperEnable := false,
       AlarmCountReset := false,  In := 0.0,    HHLimit := 0.0,     HHSeverity := 500,
       HLimit := 0.0,      HSeverity := 500,   LLimit := 0.0,      LSeverity := 500,   LLLimit := 0.0,
       LLSeverity := 500,  MinDurationPRE := 0,       Deadband := 0.0,    ROCPosLimit := 0.0,
       ROCPosSeverity := 500,     ROCNegLimit := 0.0,        ROCNegSeverity := 500,     ROCPeriod := 0.0,
       AssocTag1 := "PlantNumber",      AssocTag2 := "ShiftNumber",      AssocTag3 := "BatchNumber",
       AssocTag4 := "LotNumber",  AlarmClass := "tank2",     HMICmd := "ft command");

my_alarm : ALARM_DIGITAL (ALMMSG.AM:en-us := "my message",
       Severity := 500,    MinDurationPRE := 0,       ProgTime := DT#1970-01-01-00:00:00.000000Z,
       EnableIn := false,  In := false,  InFault := false,   Condition := true,
       AckRequired := true,       Latched := false,   ProgAck := false,   OperAck := false,
       ProgReset := false,        OperReset := false,        ProgSuppress := false      OperSuppress := false,
       ProgUnsuppress := false,   OperUnsuppress := false,   ProgDisable := false,
       OperDisable := false,      ProgEnable := false,       OperEnable := false,
       AlarmCountReset := false,  AssocTag1 := "BatchNumber",      AssocTag2 := "LotNumber",
       AssocTag3 := "PlantNumber",      AssocTag4 := "ShiftNumber",      AlarmClass := "tank1",
       HMICmd := "ft command");
```

# Define a tag-based alarm or alarm definition

## Introduction

This chapter describes the structure of tag-based alarms and alarm definitions.

A tag-based alarm is similar to a digital alarm in that it monitors a tag value to determine the alarm condition. However, a tag-based alarm is not part of the logic program and does not increase the scan time for a project. Use the New Alarm dialog box to implement a tag-based alarm. An alarm definition is associated with an Add-On Instruction (AOI) or a defined data type. When a tag is created using a data type or an AOI that has alarm definitions, alarms are created automatically based on the alarm definitions.

## Tag-based alarms

The following sections describe the tag-based alarm structure, objects, and attributes.

### L5X Tag-based alarm structure

<Tags>

<Tag Name=" pTag2" TagType="Base" Alarm="DINT" Radix="Decimal" Constant="false" ExternalAccess="Read/Write">

<AlarmConditions>

<AlarmCondition Name="Tag2HHH" Input=".1" ConditionType="HIHI" Limit="0.0" Severity="500" DelayOn="0" DelayOff="0" ShelveDuration="0" MaxShelveDuration="0" Deadband="0.0" Used="false" InFault="false" AckRequired="true" Latched="false" ProgAck="false" OperAck="false" ProgReset="false" OperReset="false" ProgSuppress="false" OperSuppress="false" ProgUnsuppress="false" OperUnsuppress="false" OperShelve="false" ProgUnshelve="false" OperUnshelve="false" ProgDisable="false" OperDisable="false" ProgEnable="false" OperEnable="false" AlarmCountReset="false" GroupOperExcluded="false" GroupRollupExcluded="false" EvaluationGroup="500 millisecond" Expression="Input = Limit" AssocTag1="dint" AssocTag2="in" AssocTag3="TotalAlarmCount" AssocTag4="TotalAlarmCount.1">

```
                    <AlarmConfig>
                  <Messages>
                   <Message Type="CAM">
                    <Text Lang="en-US">
                     <![CDATA[Message]]>
                    </Text>
                   </Message>
                  </Messages>
                  <AlarmClass>
                   <![CDATA[Class]]>
                  </AlarmClass>
                  <HMIGroup>
                   <![CDATA[FT Group]]>
                  </HMIGroup>
                  <HMICmd>
                   <![CDATA[FT View command]]>
                  </HMICmd>
                 </AlarmConfig>
                </AlarmCondition>
               </AlarmConditions>
              <Data Format="L5K">
              <![CDATA[0]]>
              </Data>
              <Data Format="Decorated">
              <DataValue DataType="DINT" Radix="Decimal" Value="0"/>
```

</Data>

</Tag>

</Tags>

## L5K Tag-based alarm structure

Each alarm starts with ALARM_CONDITION and ends with END_ALARM_CONDITION. The alarm definitions location in the L5K file will be after the tasks. The example below is an alarm that was based on an alarm definition.

ALARM_CONDITION PVHH (Name := "PVHH",

AlarmConditionDefinition := "PVHH",

Input := "test.PVFault",

ConditionType := "TRIP",

Limit := 0.0,

Severity := 500,

OnDelay := 500,

OffDelay := 1000,

DefaultShelveDuration := 0,

MaxShelveDuration := 0,

Deadband := 0.0,

Used := true,

InFault := false,

AckRequired := true,

Latched := false,

ProgAck := false,

OperAck := false,

ProgReset := false,

```
                                        OperReset := false,
                                        ProgSuppress := false,
                                        OperSuppress := false,
                                        ProgUnsuppress := false,
                                        OperUnsuppress := false,
                                        OperShelve := false,
                                        ProgUnshelve := false,
                                        OperUnshelve := false,
                                        ProgDisable := false,
                                        OperDisable := false,
                                        ProgEnable := false,
                                        OperEnable := false,
                                        AlarmCountReset := false,
                                        AlarmSetOperIncluded := false,
                                        AlarmSetRollupIncluded := false,
                                        EvaluationPeriod := 2,
                                        Expression := 19,
                                        AssocTag1 := "test.EnableIn",
                                        AssocTag2 := "test.SPHLimit",
                                        AssocTag3 := "test.SPLLimit")
                                END_ALARM_CONDITION
```

The following example is for an alarm that is not based on an alarm definition and has one localized message.

```
ALARM_CONDITION HH (ALMMSG.CAM:zh-TW := "mmm",
                                        Name := "HH",
```

```
Input := "\MainProgram.bool",
ConditionType := "TRIP",
Limit := 0.0,
Severity := 500,
OnDelay := 0,
OffDelay := 0,
DefaultShelveDuration := 0,
MaxShelveDuration := 0,
Deadband := 0.0,
Used := true,
InFault := false,
AckRequired := true,
Latched := false,
ProgAck := false,
OperAck := false,
ProgReset := false,
OperReset := false,
ProgSuppress := false,
OperSuppress := false,
ProgUnsuppress := false,
OperUnsuppress := false,
OperShelve := false,
ProgUnshelve := false,
OperUnshelve := false,
ProgDisable := false,
```

```
OperDisable := false,
ProgEnable := false,
OperEnable := false,
AlarmCountReset := false,
AlarmSetOperIncluded := true,
AlarmSetRollupIncluded := true,
EvaluationPeriod := 2,
Expression := 128,
AlarmClass := "aaa",
HMICmd := "ccc")
END_ALARM_CONDITION
```

## Tag-based alarm objects

This section describes tag-based alarm objects and their attributes.

**Alarm Conditions**

Alarm conditions are a collection of the alarms for a tag.

- AlarmCondition -- An alarm condition in this collection for this tag.

**Alarm Condition**

The alarm condition on a tag. The following table lists Alarm Condition attributes.

| Programmatic Name | Attribute Description |
|---|---|
| Name | The name of the alarm. |
| Input | Fully qualified path of a Scalar Tag Item that has this alarm. |
| Condition | The alarm condition type. See table below for values and the relation to the input data type. |
| AlarmConditionDefinition | The alarm condition definition that this alarm is based on. |
| Evaluation Period | How frequently the alarm limit will be evaluated.<br>There is only one evaluation period and it is 500. |
| Expression | The expression for evaluating the condition compared to the limit. See the table below for the expression for the various condition types. |

| | |
|---|---|
| Limit | Alarm limit.<br>Default = 0.0. |
| TargetTag | Tag used to in the condition's expression for input datatypes that are not BOOL or BIT. |
| AssocTag1<br>AssocTag2<br>AssocTag3<br>AssocTag4 | Up to four specified tags that contain pertinent data when alarm conditions occur. A snapshot of the values in these tags is stored whenever the status changes to the In Alarm state, or other alarm state changes occur. The stored values may also be embedded in the alarm Message. |
| Severity | Severity of the High alarm condition. This does not affect processing of alarms by the controller, but can be used for sorting and filtering functions at the alarm subscriber.<br>Valid = 1...1000 (1000 = most severe; 1 = least severe).<br>Default = 500. |
| OnDelay | Duration (milliseconds) for the alarm condition to remain true before the alarm is marked as InAlarm and alarm notification is sent to clients. The controller collects alarm data as soon as the alarm condition is detected, so no data is lost while waiting to meet the minimum duration.<br>Valid = 0...2147483647.<br>Default = 0. |
| OffDelay | Duration (milliseconds) for the alarm condition to remain false before the alarm is marked as not InAlarm and alarm notification is sent to clients. The controller collects alarm data as soon as the alarm condition is detected, so no data is lost while waiting to meet the minimum duration.<br>Valid = 0...2147483647.<br>Default = 0. |
| ShelveDuration | Time duration (in minutes) for which a shelved alarm will be shelved. Minimum time is one minute. Maximum time is defined by MaxShelveDuration. |
| MaxShelveDuration | Maximum time duration (in minutes) for which an alarm can be shelved. |
| Deadband | Deadband for detecting when the alarm levels have returned to normal.<br>A non-zero Deadband can reduce alarm condition chattering if the In value is continually changing but remaining near the level condition threshold. The Deadband value does not affect the transition to the InAlarm (active) state.<br>Default = 0.0. |
| InFault | Bad health indicator for the input. The user application may set InFault to indicate the input signal has an error. When set, the instruction sets InFaulted (Status.1). When cleared, the instruction clears InFaulted (Status.1). In either case, the instruction continues to evaluate In for alarm conditions.<br>Default is cleared (good health). |
| AckRequired | Specifies whether alarm acknowledgment is required. When set, acknowledgment is required. When cleared, acknowledgment is not required and Acked is always set.<br>Default is set. |
| Latched | Specifies whether the alarm is latched. Latched alarms remain InAlarm when the alarm condition becomes false, until a Reset command is received. When set, the alarm is latched. When cleared, the alarm is unlatched.<br>Default is cleared.<br>A latched alarm can only be reset when the alarm condition is false.<br>Default is cleared. |
| ProgAck | Set by the user program to acknowledge the alarm. Takes effect only if the alarm is unacknowledged. Requires a cleared-to-set transition. |
| OperAck | Set by the operator interface to acknowledge the alarm. Takes effect only if the alarm is unacknowledged. The instruction clears this parameter.<br>Default is cleared. |
| ProgReset | Set by the user program to reset the latched alarm. Takes effect only if the latched alarm is InAlarm and the alarm condition is false. Requires a cleared-to-set transition. |
| OperReset | Set by the operator interface to reset the latched alarm. Takes effect only if the latched alarm is InAlarm and the alarm condition is false. The alarm instruction clears this parameter.<br>Default is cleared. |

| | |
|---|---|
| ProgSuppress | Set by the user program to suppress the alarm.<br>Default is cleared. |
| OperSuppress | Set by the operator interface to suppress the alarm. The alarm instruction clears this parameter.<br>Default is cleared. |
| ProgUnsuppress | Set by the user program to unsuppress the alarm. Takes precedence over Suppress commands.<br>Default is cleared. |
| OperUnsuppress | Set by the operator interface to suppress the alarm. The alarm instruction clears this parameter.<br>Default is cleared. |
| OperShelve | Set by the operator interface to shelve or reshelve the alarm. Requires a transition from a cleared state in one program scan to a set state in the next program scan. The alarm instruction clears this parameter. Default is cleared.<br>Unshelve commands take precedence over Shelve commands.<br>Shelving an alarm postpones alarm processing. It is like suppressing an alarm, except that shelving is time-limited. If an alarm is acknowledged while it is shelved, it remains acknowledged even if it becomes active again. It becomes unacknowledged when the shelve duration ends provided the alarm is still active at that moment. |
| ProgUnshelve | Set by the user program to unshelve the alarm. Takes precedence over Shelve commands. Default is cleared.<br>For more information on shelving an alarm, see the description for the OperShelve parameter. |
| OperUnshelve | Set by the operator interface to unshelve the alarm. The alarm instruction clears this parameter. Takes precedence over Shelve commands. Default is cleared.<br>For more information on shelving an alarm, see the description for the OperShelve parameter. |
| ProgDisable | Set by the user program to disable the alarm.<br>Default is cleared. |
| OperDisable | Set by the operator interface to disable the alarm. The alarm instruction clears this parameter.<br>Default is cleared. |
| ProgEnable | Set by the user program to enable the alarm. Takes precedence over a Disable command. |
| OperEnable | Set by the operator interface to enable the alarm. Takes precedence over Disable command. The alarm instruction clears this parameter.<br>Default is cleared. |
| AlarmCountReset | Set by the operator interface to reset the alarm count to zero. The alarm instruction clears this parameter. Default is cleared. |
| Used | Indicates whether the alarm instance is used or not:<br>True (1) – Alarm instance is used<br>False (0) – Alarm instance is not used |
| AlarmSetOperIncluded | Boolean opting in or out of having any operation done on the alarm set is also done on this alarm. |
| AlarmSetRollupIncluded | Boolean opting in or out of whether this alarm is included in the alarm sets' count. |
| AlarmConfig | Additional Configuration for this alarm. Includes the following four elements:<br>• AlarmMessages -- Messages that are displayed when the alarm is activated.<br>• AlarmClass -- String grouping together related alarms.<br>• HMICmd -- Command string that gets executed on the HMI when the alarm is activated.<br>• HMIGroup -- String grouping together related alarm for the HMI. |

The following table lists alarm condition types.

| Input Data Type | Condition Type | Example Expression |
|---|---|---|
| BOOL/BIT | TRIP | Input=0 |
| BOOL/BIT | HIHI<br>HI<br>LO<br>LOLO<br>DEV_HI<br>DEV_LO<br>ROC_POS<br>ROC_NEG | Input=0 |
| Numeric (DINT, REAL, etc.) | HIHI<br>HI | Input>Limit |
| Numeric (DINT, REAL, etc.) | LO<br>LOLO | Input<Limit |
| Numeric (DINT, REAL, etc.) | LOLO | Input>Limit |
| Numeric (DINT, REAL, etc.) | ROC_POS | Input >= Limit |
| Numeric (DINT, REAL, etc.) | ROC_NEG | Input <= Limit |
| Numeric (DINT, REAL, etc.) | DEV_HI | Input >= TargetTag + Limit |
| Numeric (DINT, REAL, etc.) | DEV_LO | Input <= TargetTag - Limit |

### AlarmConfig

Provides the following additional attributes:

- AlarmMessages -- Messages that are displayed when the alarm is activated.
- AlarmClass -- String grouping together related alarms.
- HMICmd -- Command string that gets executed on the HMI when the alarm is activated.
- HMIGroup -- String grouping together related alarm for the HMI.

### AlarmMessages

Contains the messages for the alarm.

### AlarmMessage

A message for the alarm. Includes the following attributes:

- Type -- Must be CAM.
- Text -- The text of the alarm message.

**Text**

The text of the alarm message in a language. Includes the following attributes:

- Lang -- Language for the alarm message.
- Value -- Content of the alarm message.

## Tag-based alarm definitions

The following sections describe the tag-based alarm definition structure, objects, and attributes.

### L5X Tag-based alarm definition structure

```
<AlarmDefinitions>
  <DataTypeAlarmDefinition Name="PID_ENHANCED">
<MemberAlarmDefinition Name="HH" Input=".SP" ConditionType="HIHI"
Limit="0.0" Severity="500" DelayOn="0" DelayOff="0" ShelveDuration="0"
MaxShelveDuration="0" Deadband="0.0" Required="false" InFault="false"
AckRequired="true" Latched="false" GroupOperExcluded="false"
GroupRollupExcluded="false" EvaluationGroup="500 millisecond"
Expression="Input = Limit" AssocTag1=".PV" AssocTag2=".CV" >
    <AlarmConfig>
     <Messages>
      <Message Type="ADM">
       <Text Lang="en-US">
        <![CDATA[Message]]>
       </Text>
      </Message>
     </Messages>
     <AlarmClass>
      <![CDATA[Class]]>
     </AlarmClass>
```

```
        <HMIGroup>
         <![CDATA[FT Group]]>
        </HMIGroup>
        <HMICmd>
         <![CDATA[FT View command]]>
        </HMICmd>
       </AlarmConfig>
            </MemberAlarmDefinition >
      </DataTypeAlarmDefinition >
</AlarmDefinitions>
<Tags>
```

## L5K Tag-based alarm definition structure

```
ALARM_DEFINITION PVHH (ALMMSG.ADM:en-US := "PIDE Message 3",
                       ALMMSG.ADM:zh-TW := "This is my message",
                       Name := "PVHH",
                       Input := "PID_ENHANCED.PVFault",
                       ConditionType := "TRIP",
                       Limit := 0.0,
                       Severity := 500,
                       OnDelay := 500,
                       OffDelay := 1000,
                       DefaultShelveDuration := 0,
                       MaxShelveDuration := 0,
                       Deadband := 0.0,
```

```
                                        Required := true,
                                        InFault := false,
                                        AckRequired := true,
                                        Latched := false,
                                        AlarmSetOperIncluded := false,
                                        AlarmSetRollupIncluded := false,
                                        EvaluationPeriod := 2,
                                        Expression := 19,
                                        AssocTag1 :=
"PID_ENHANCED.EnableIn",
                                        AssocTag2 :=
"PID_ENHANCED.SPHLimit",
                                        AssocTag3 :=
"PID_ENHANCED.SPLLimit",
                                        AlarmClass := "My Class",
                                        HMICmd := "My FT View
Command")
            END_ALARM_DEFINITION
```

## Tag-based alarm definition objects

This section describes tag-based alarm definition objects and their attributes.

**AlarmDefinitions**

The collection of all of the alarm definitions in the project.

- DatatypeAlarmDefinition -- The alarm definitions for a specific datatype.

**DataTypeAlarmDefinition**

The collection of the alarm definitions for a specific datatype.

- Name -- The name of the datatype that these alarm definitions are for.

- MemberAlarmDefinition -- The alarm definitions for a specific member of this datatype.

### MemberAlarmDefinition

The alarm definition for a specific member of a datatype. The following table describes the alarm definition attributes.

| Programmatic Name | Attribute Description |
|---|---|
| Name | The name of the alarm definition. |
| Input | Fully qualified path of a datatype member that this alarm definition is on. |
| ConditionType | The alarm condition type. |
| Evaluation Period | How frequently the alarm limit will be evaluated.<br>There is only one evaluation period and it is 500. |
| Expression | The expression for evaluating the condition compared to the limit. |
| Limit | Alarm limit.<br>Default = 0.0. |
| TargetTag | Tag used to in the condition's expression for input datatypes that are not BOOL or BIT. |
| AssocTag1<br>AssocTag2<br>AssocTag3<br>AssocTag4 | Up to four specified tags that contain pertinent data when alarm conditions occur. A snapshot of the values in these tags is stored whenever the status changes to the In Alarm state, or other alarm state changes occur. The stored values may also be embedded in the alarm Message. |
| Severity | Severity of the High alarm condition. This does not affect processing of alarms by the controller, but can be used for sorting and filtering functions at the alarm subscriber.<br>Valid = 1...1000 (1000 = most severe; 1 = least severe).<br>Default = 500. |
| OnDelay | Duration (milliseconds) for the alarm condition to remain true before the alarm is marked as InAlarm and alarm notification is sent to clients. The controller collects alarm data as soon as the alarm condition is detected; so no data is lost while waiting to meet the minimum duration.<br>Valid = 0...2147483647.<br>Default = 0. |
| OffDelay | Duration (milliseconds) for the alarm condition to remain false before the alarm is marked as not InAlarm and alarm notification is sent to clients. The controller collects alarm data as soon as the alarm condition is detected; so no data is lost while waiting to meet the minimum duration.<br>Valid = 0...2147483647.<br>Default = 0. |
| ShelveDuration | Time duration (in minutes) for which a shelved alarm will be shelved. Minimum time is one minute. Maximum time is defined by MaxShelveDuration. |
| MaxShelveDuration | Maximum time duration (in minutes) for which an alarm can be shelved. |
| Deadband | Deadband for detecting when the alarm levels have returned to normal.<br>A non-zero Deadband can reduce alarm condition chattering if the In value is continually changing but remaining near the level condition threshold. The Deadband value does not affect the transition to the InAlarm (active) state.<br>Default = 0.0. |
| AckRequired | Specifies whether alarm acknowledgment is required. When set, acknowledgment is required. When cleared, acknowledgment is not required and Acked is always set.<br>Default is set. |

| | |
|---|---|
| Latched | Specifies whether the alarm is latched. Latched alarms remain InAlarm when the alarm condition becomes false, until a Reset command is received. When set, the alarm is latched. When cleared, the alarm is unlatched.<br>Default is cleared.<br>A latched alarm can only be reset when the alarm condition is false.<br>Default is cleared. |
| Required | Whether or not instances of this must always be marked as used |
| AlarmSetOperIncluded | Boolean opting in or out of having any operation done on the alarm set is also done on this alarm. |
| AlarmSetRollupIncluded | Boolean opting in or out of whether this alarm is included in the alarm sets' count. |
| AlarmConfig | Additional Configuration for this alarm. Includes the following elements:<br>• AlarmMessages -- Messages that are displayed when the alarm is activated.<br>• AlarmClass -- String grouping together related alarms.<br>• HMICmd -- Command string that gets executed on the HMI when the alarm is activated.<br>• HMIGroup -- String grouping together related alarm for the HMI. |
| Messages | The messages for this alarm definition. |

# Define a program component

## Introduction

This chapter explains the overall structure of the Program component.

## Program component

The Program component defines the programs used in the logic you export.

The maximum number of programs depends on the type of controller.

| Controller | Maximum Number of Programs |
|---|---|
| ControlLogix | 1000<br>(32 in firmware revisions prior to 15) |
| SoftLogix 5800 | 100<br>(32 in firmware revisions prior to 15) |
| FlexLogix | 32 |
| CompactLogix | 32 |
| DriveLogix | 32 |

## L5X program structure

```
<Programs>
      <Program [Program_Attributes]>
            <Description>
                  <![CDATA[ text ]]  >
            </Description>
            <Tags>
                  tags
            </Tags>
            <Routines>
                  routines
            </Routines>
            <ChildPrograms>
                  child_programs
            </ChildPrograms>
      </Program>
</Programs>
```

## L5K PROGRAM structure

```
PROGRAM <program_name> [(Description := "text",
      Program_Attributes)]
      [TAG declaration]
      [ROUTINE declaration]
      [FBD_ROUTINE declaration]
      [SFC_ROUTINE declaration]
      [ST_ROUTINE declaration]
      [CHILD_PROGRAMS]
END_PROGRAM
```

## Program elements

| L5X Item | L5K Item | Description |
|---|---|---|
| Not applicable | *program_name* | The name of the program.<br><br>In L5X, use a Name attribute on the <Program> element. |
| Description | Description | User information about the program. |
| *tags* | TAG | Program-scoped tags.<br>Follows same format as controller-scoped tags.<br>For more details on defining a tag, see Chapter 6 Defining a Tag Component on page 107. |
| *routines* | ROUTINE<br>FBD_ROUTINE<br>ST_ROUTINE<br>SFC_ROUTINE | Routines within the program.<br>For more details on defining a:<br>• Ladder logic routine, see Chapter 8 Defining a Ladder Logic Routine on page 167.<br>• Function block diagram routine, see Chapter 9 defining a Function Block Diagram Routine on page 179.<br>• Sequential function chart routine, see Chapter 10 Defining a Sequential Function Chart Routine on page 201.<br>• Structured text routine, see Chapter 11 Defining a Structured Text Routine on page 221. |
| *ChildPrograms* | CHILD_PROGRAMS | Child programs of the parent program. The child-parent relationship is visible in only the Logical Organizer in the Logix Designer application, and not in the Controller Organizer. |

## Program attributes

| Attribute | Description |
|---|---|
| Name | L5X only. Specify the name of the program.<br><br>In L5K, the name is an element of the statement. |
| Type | Specify the type of program. If this program is a Equipment Phase program, type **EquipmentPhase**. If this program is a Sequence program, type **Sequence**. Otherwise, type **Normal**. |
| Class | Specify the class of the program. This attribute applies only to safety controller projects. Type **Standard** or **Safety**. Do not use this attribute if the program is an Equipment Phase program.<br>(Type = EquipmentPhase). |

| Attribute | Description |
|---|---|
| Main (L5K)<br>MainRoutineName (L5X) | Specify the name of the main routine of the program (40 characters maximum). |
| Fault (L5K)<br>FaultRoutineName (L5X) | L5K only. Specify the name of the program fault routine, if any (40 characters maximum). |
| Mode | L5K only. Type **0** for not testing edits; type **1** for testing edits. |
| TestEdits | L5X only. Type **false** for not testing edits; type **true** for testing edits. |
| DisableFlag | L5K only. Type **1** to disable the program; type **0** to enable the program. |
| Disabled | L5X only. Type **true** to disable the program; type **false** to enable the program. |
| SynchronizeRedundancyData AfterExecution | Type **1** to synchronize data after the program scan in a redundant system; type **0** to not synchronize data after the program scan. |
| UseAsFolder | Type **1** to use the program as a folder. Logix Designer application does not execute any routines or update any parameters in a program that is used as a folder. |

## Program attributes for EquipmentPhase programs

In addition to the program connection attributes previously described, an Equipment Phase program (Program Type = EquipmentPhase) has the following attributes.

| Attribute | Description |
|---|---|
| PreState (L5K)<br>PreStateRoutineName (L5X) | L5K only. Specify the name of the prestate routine (40 characters maximum). |
| InitialStepIndex | Specify an integer value for the initial step index of the phase. |
| InitialState | Specify state of the phase. Type Aborted, Completed, Stopped, or Idle (default). |
| CompleteStateIfNotImpl | If the phase does not implement all the expected states, enter StateComplete (default) so the program can continue to execute when it expects a state that was omitted. The program ignores the omitted state and continues to the next state. Otherwise, enter NoAction. |
| LossOfCommCmd | If the phase uses an external sequencer, such as FactoryTalk Batch® software, specify that appropriate action to take if communication fails between the controller and the external sequencer. Type **Abort**, **Hold**, **Stop**, or **none** (default). |
| ExternalRequestAction | Specify how to handle an external request (PXRQ instruction) that is in process when the phase receives the command to go to a Holding state. Type **Clear** to abort outstanding external requests. Otherwise, type **none** (default). |
| EquipmentId | The FactoryTalk Batch equipment identifier for the Equipment Phase. This value is set by the FactoryTalk Batch software when you synchronize with a project file. Do not modify this value. |
| RecipePhaseNames | The FactoryTalk Batch recipe phases for the Equipment Phase. This value is set by the FactoryTalk Batch software when you synchronize with a project file. Do not modify this value. |
| AutoValueAssignStepToPhase | Specify whether the value of sequence step inputs are automatically assigned to phase inputs when a sequence step starts the phase and the phase is used in a sequence.<br><br>In L5K, type **Yes** or **No**. In L5X, type **true** or **false**. |
| AutoValueAssignPhaseToStepOnComplete | Specify whether the value of phase outputs are automatically assigned to step outputs when the phase is completed and the phase is used in a sequence.<br><br>In L5K, type **Yes** or **No**. In L5X, type **true** or **false**. |

| Attribute | Description |
|---|---|
| AutoValueAssignPhaseToStepOnStopped | Specify whether the value of phase outputs are automatically assigned to step outputs when the phase is stopped and the phase is used in a sequence.<br><br>In L5K, type **Yes** or **No**. In L5X, type **true** or **false**. |
| AutoValueAssignPhaseToStepOnAborted | Specify whether the value of phase outputs are automatically assigned to step outputs when the phase is stopped and the phase is used in a sequence.<br><br>In L5K, type **Yes** or **No**. In L5X, type **true** or **false**. |

## Program attributes for Sequence programs

In addition to the program connection attributes previously described, a Sequence program (Program Type = Sequence) has the following attributes.

| Attribute | Description |
|---|---|
| Revision | Revision number for the program, consisting of major and minor revision. |
| RevisionExtension | Additional revision information. |
| UnitID | Numeric identifier to designate the equipment that the sequence is controlling.<br><br>Example value: 1-0x7FFFFFFF |
| RetainSequenceIDOnReset | Specify whether the sequence ID is retained when the sequence resets.<br><br>In L5K, type **Yes** or **No**. In L5X, type **true** or **false**. |
| GenerateSequenceEvents | Specify whether the sequence events are generated.<br><br>In L5K, type **Yes** or **No**. In L5X, type **true** or **false**. |
| ValuesToUseOnStart | Specify the values of sequence parameters and step tags that are used when starting the sequence.<br>• **Use Initial Values** – Copy the initial values to the current values when starting the sequence.<br>• **Retain Current Values** – Use the current values of sequence parameters and step tags when starting the sequence. |
| ValuesToUseOnReset | Specify the how the values of sequence parameters and step tags are set when the sequence resets.<br>• **Use Initial Values** – Restore the current values of sequence parameters and step tags to the initial values when resetting the sequence.<br>• **Retain Current Values** – Maintain the current values of sequence parameters and step tags when resetting the sequence. |
| RevisionNote | Optional detailed information about the revision.<br>For L5X, this is a sub element: <RevisionNote>. |

## Child program component

The Child Program component defines the child programs used in the logic you export. A child program is a program defined as the logical child of another program in the Logical Organizer in the Logix Designer.

## L5K CHILD_PROGRAM structure

```
PROGRAM <program_name> [(Description := "text",
      Program_Attributes)]
      [TAG declaration]
      [ROUTINE declaration]
      [FBD_ROUTINE declaration]
      [SFC_ROUTINE declaration]
      [ST_ROUTINE declaration]
      [CHILD_PROGRAMS]
END_PROGRAM
```

## L5X child program structure

```
<Programs>
      <Program [Program_Attributes]>
            <Description>
                  <![CDATA[ text ]]  >
            </Description>
            <Tags>
                  tags
            </Tags>
            <Routines>
                  routines
            </Routines>
            <ChildPrograms>
                  child_programs
            </ChildPrograms>
      </Program>
</Programs>
```

## Child program attributes

| Attribute | Description |
|---|---|
| child_program_name | L5X only. Specify the name of the child program.<br><br>In L5K, the name is an element of the statement. |

## Child program guidelines

Observe these guidelines when defining a child program.

- Since child programs are a way to organize existing programs, adding a child program component does not add to the number of programs.

# Encoded/Unencoded routines

These examples are for protected (encoded) and unprotected (clear text) codes for routines.

If the project contains a source-protected routine, the routine appears within an encoded data component in the program. See Exporting Source-protected Logic on page 28 on page 34 for procedures.

## L5X EncodedData Structure

```
<EncodedData EncodedType= "type", Name="name",
Type="routinetype"
            [,other_attributes]>
      <Description>
            <![CDATA[ text ]] >
      </Description>
      encoded_data
</EncodedData>
```

## L5K ENCODED_DATA Structure

```
ENCODED_DATA [( EncodedType: type, Name:= name,
              Type:= routinetype,
              other_attributes)]
      encoded_data
END_ENCODED_DATA
```

## Encoded Data Elements

| L5X Item | L5K Item | Description |
|---|---|---|
| *type* | *type* | The type of data encoded.<br><br>For L5K: ROUTINE<br>For L5X: Routine |
| *name* | *name* | The name of the protected routine. |
| *routinetype* | *routinetype* | The type of the routine protected (RLL, FBD, SFC, or ST). |
| *other_attributes* | *other_attributes* | Attributes of the routine that are not protected during export. |
| *encoded_data* | *encoded_data* | The protected portion of the routine. |

**Important:** When the routine is source protected, the *encoded_data* information is encrypted. If you modify this encrypted information, you will not be able to re-import the routine.

## Encoded Information elements

| L5X Item | L5K Item | Identifies |
|---|---|---|
| EncryptionKey | ENCRYPTION_KEY | Identifies the options the user has chosen for protecting and locking their content. |

## Encoded key attributes

| Attribute | Description |
|---|---|
| Name | Identifies the type of protection. |
| ID | Identifier for the key (Firm code, product code). |
| Description | Description of what the key is associated with (license name). |
| Vendor | Indicates the vendor who supplied the key. |
| PublicKey | Stores the public key that will be used for the locking of the associated object. |

## Encoded content attributes

| Attribute | Description |
|---|---|
| EncryptedType | Indicates the underlying language of the routine for this encoded content (for example, RLL or Structured text). |
| OnlineEditType | L5X only. Specify the online edit logic type (**Original**, **PendingEdits**, or **TestEdits**). This attribute is not specified if there are no edits. |

## L5K source protected routine example

```
ENCODED_DATA (EncodedType := ROUTINE,
              Name := "ProcessControl",
              Description := "description on routine",
              Type := SFC,
              EncryptionConfig := 2)

17r8GxtsZCMLfk3JHFYmU7emZMNhRh9oEuuPQb5IKS676d/xRznQ+56vf8IVQNNEIDoDL1u+UEC3OIMDetvnJAX2Cdw
NPRnC1N3cjPApchCL9ShdF1y2x3/T47RDSI1z99blXN5v5xUQTG1eVktB6dSpatujrGLLTf4mOEdFvHmD9qQTNep+e/
M9V9Cx5Iz2s4xc3Rll3OF9KKobr7j1RdDmAuyRRZLyqKTE01zDpZAS9AlcebPoYUhM1gCF1AuK9eYBkzm/VI5gpH4Oc
vUci/xdt0/9/xPPXkiZjoDI2ZoPFdkZ74VOGrL43wK0NQX2wPc/u4RCCDehNMxBdSYffba1UIvy9FDrJwxOd/k8wrlr
6BW0kLeBylVGxCuCxa8OYersEhBLz6nrTYd/wybHvNmp0TLgpI1bhzwyTVttwf4ZW8qBN6Yu762cUwba/btkCQ40Uic
6Mit6vMt/TFUR45J3Hw/dwGPKUaxDBNBbbBvDB11FncORX/Mtm+HAyqF5vtt6LWKwoZaEvI76sojhC8mNJ0LuBdPtZc
LafqSF2OMbwedr9CW6w
                END_ENCODED_DATA
```

## L5X source-protected routine examples

```
- <EncodedData EncodedType="Routine" Name="ProcessControl" Type="SFC" EncryptionConfig="2">
  - <Description>
      <![CDATA[ description on routine  ]]>
    </Description>

    uLqFGyBsViM1fiLJP1YXU4ymWcNNRlBoLOvAQfNIey7w6ZzxFjmz+5Svf8ICQM5EKzoGLxW
  </EncodedData>
```

## Program guidelines

Observe these guidelines when defining a program.

- Define the main and fault attributes in any order.
- You must put the tag declaration block before the routine block.

- Define up to 1000 programs per task.
- When you import a project:
  - All tag collection declaration blocks in a program definition block are imported as local tags of a given program and are seen only by routines under that program.
  - Program parameters are imported as parameters of a given program, but are seen by other tags, parameters, and routines in the project.
  - Controller tags are seen by routines in any program.

## Examples

**L5X Program Example**

```
- <Program Name="MainProgram" TestEdits="false" MainRoutineName="MainRoutine"
    Disabled="false" UseAsFolder="false">
  - <Tags>
    - <Tag Name="Test_CurrentDate" TagType="Base" DataType="BOOL"
        Radix="Decimal">
        <Data>00</Data>
      - <Data Format="Decorated">
          <DataValue DataType="BOOL" Radix="Decimal" Value="0" />
        </Data>
      </Tag>
    </Tags>
  - <Routines>
    - <Routine Name="MainRoutine" Type="RLL">
      - <RLLContent>
        - <Rung Number="0" Type="N">
          - <Comment>
              <![CDATA[ Get the Controllers real time clock broken
              down by Year, month, day...microseconds and store in a tag so that the valu
            </Comment>
          - <Text>
              <![CDATA[ XIC(Test_CurrentDate)GSV
              (WALLCLOCKTIME,,DateTime,TestDateTime.Year);  ]]>
            </Text>
          </Rung>
        </RLLContent>
      </Routine>
    </Routines>
    <ChildPrograms>
      <ChildProgram Name="Prog01002" />
    </ChildPrograms>
  </Program>
```

**L5K PROGRAM Example**

```
PROGRAM MainProgram (MAIN := "MainRoutine",
                     MODE := 0,
                     DisableFlag := 0,
                     UseAsFolder  := 0)
                TAG
                        Test_CurrentDate : BOOL (RADIX := Decimal) := 0;
                END_TAG

                ROUTINE MainRoutine
                                    RC: "Get the Controllers real time clock broken down
by Year, month, day...microseconds and store in a tag so that the values can be used.";
                                    N: XIC(Test_CurrentDate)GSV
(WALLCLOCKTIME,,DateTime,TestDateTime.Year);
                END_ROUTINE
                CHILD_PROGRAMS
                    Program010010
                END_CHILD_PROGRAMS
PROGRAM END_PROGRAM
```

**L5X Equipment Phase Program Example**

```
- <Program Name="Add_Cream_M2" Type="EquipmentPhase" TestEdits="false"
    Disabled="false" InitialStepIndex="0" InitialState="Idle"
    CompleteStateIfNotImpl="StateComplete" LossOfCommCmd="None"
    ExternalRequestAction="None" UseAsFolder="false">
  - <Tags>
    - <Tag Name="MyCounter" TagType="Base" DataType="DINT" Radix="Decimal">
        <Data>00 00 00 00</Data>
      - <Data Format="Decorated">
          <DataValue DataType="DINT" Radix="Decimal" Value="0" />
        </Data>
      </Tag>
    </Tags>
  - <Routines>
    + <Routine Name="Running" Type="ST">
    </Routines>
    <ChildPrograms>
      <ChildProgram Name="Prog01002" />
    </ChildPrograms>
  </Program>
```

**L5K Equipment Phase PROGRAM Example**

```
PROGRAM Add_Cream_M2 (Type := EquipmentPhase,
                      MODE := 0,
                      DisableFlag := 0,
                      InitialStepIndex := 0,
                      InitialState := Idle,
                      CompleteStateIfNotImpl := StateComplete,
                      LossOfCommCmd := None,
                      ExternalRequestAction := None
                      UseAsFolder  := 0)
                    TAG
                        MyCounter : DINT (RADIX := Decimal) := 0;
                    END_TAG

                    ST_ROUTINE Running
                            '//
                            '// All information provided "AS IS" -- No warranty or implied
merchantability.
                            '// Refer to the RSLogix 5000 End User License Agreement
(EULA) in the Release Notes.

                            '//
                            '
                            'MyCounter := MyCounter + 1;
                            'if (MyCounter > MAX_COUNT) then
                            '       PSC();
                            '       MyCounter := 0;
                            'end_if;
                            '
                    END_ST_ROUTINE
                    CHILD_PROGRAMS
                        Program010010
                    END_CHILD_PROGRAMS
END_PROGRAM
```

# Define a ladder logic routine

## Introduction

This chapter explains how to enter ladder diagram logic in a complete import/export file.

## Ladder logic routine

These examples show the ladder logic routine structure.

### L5X ladder logic routine structure

```
<Routines>
      <Routine [Routine_Attributes]>
            <Description>
                  <![CDATA[ text ]]  >
            </Description>
            <RLLContent>
                  logic
            </RLLContent>
      </Routine>
</Routines>
```

### L5K Ladder Logic ROUTINE structure

```
ROUTINE <routine_name> [(Description := "text")]
      <ladder_rungs>
END_ROUTINE
```

### Ladder logic routine elements

| L5X Item | L5K Item | Description |
|---|---|---|
| N/A | *routine_name* | The name of the routine.<br><br>In L5X, use a Name attribute in the <Routine> element. |
| Description | Description | User information about the routine. |
| RLLContent | *ladder_rungs* | Rung logic. |
| EncryptionInfo | ENCRYPTION_INFO | Details of the license-based Source Protection for the lockable object. Only exists for protected routines exported in plain text. |
| EncryptedContent | N/A | Source Protected and locked routine content. Only exists for locked routines exported in plain text. |
| EncryptedSegments | N/A | Locked logic for the routine. Only exists for locked routines exported in plain text. |

## RLL Routine attributes

| Attribute | Description |
|---|---|
| Name | L5X only. Specify the name of the routine.<br><br>In L5K, the name is an element of the statement. |
| Type | L5X only. Specify RLL.<br><br>In L5K, the type of routine is part of the routine statement. |
| PermissionSet | Name of the set of permissions, configured in FactoryTalk Security, to apply to this object. |
| TrackingGroups | The group of tracked objects to which this item belongs. Components can be marked for tracking to determine whether they have been changed. Version 30 of the Logix Designer application supports only one tracking group. |

# Rung logic

Enter rung logic within a routine component in an import/export file.

## L5X rung structure

```
<RLLContent>
      <Rung [Rung_Attributes]>
            <Comment>
                  <![CDATA[ comment_text ]] >
            </Comment>
            <Text >
                  <![CDATA[ rung_neutral_text; ] ] >
            </Text>
      </Rung>
</RLLContent>
```

## L5K RUNG structure

```
RC: "comment" "more" "etc";
<RungType> : <rung_neutral_text>;
```

## Rung elements

| L5X Item | L5K Item | Description |
|---|---|---|
| N/A | RungType | The type of rung.<br><br>In L5X, use a Type attribute on the <Rung> element. |
| Comment | RC | Rung comment.<br><br>In a L5K file, a rung comment must be followed by a rung. |
| Text | *rung_neutral_text* | The logic. |

These rung types are available.

| Item | Identifies |
|---|---|
| N | Normal |
| I | Insert |
| D | Delete |

| Item | Identifies |
|---|---|
| IR | Insert with a replace |
| rR | Pending replace IR |
| R | Replace |
| rI | Pending replace I |
| rN | Pending replace N |
| e | Pending insert rung |
| er | Pending replace rung |
| d | Pending delete rung |

## Rung attributes

| Attribute | Description |
|---|---|
| Number | L5X only. Specify the rung number within the routine.<br><br>In L5K, the rung number is defined by the order the rung appears in the L5K format. |
| Type | L5X only. Specify RLL.<br><br>In L5K, the type of rung is part of the rung statement. |

# Rung guidelines

Use these guidelines for rung logic:

- Rungs are specified by using neutral language. See Neutral Text for Ladder Instructions on page 156 on page 171 for the format for the supported instructions.
- Each rung ends with a semicolon (;)

# Branches

Enter a single branch or simultaneous branches on a rung.

## L5X branch structure

```
<Branch [BranchAttributes]>
      <Leg [LegAttributes]/>
</Branch>
```

## L5K BRANCH structure

```
BRANCH (Branch_Attributes)
      LEG (Leg_Attributes)
      END_LEG
END_BRANCH
```

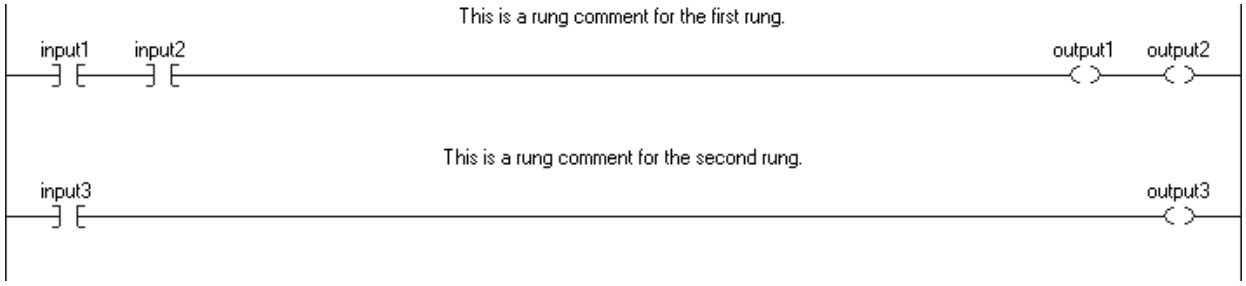
## L5X Examples

**L5X Ladder ROUTINE Example**

```
- <Routine Name="Test_DOW" Type="RLL">
  - <RLLContent>
    - <Rung Number="0" Type="N">
      - <Comment>
        + <![CDATA[  ]]>
        </Comment>
      - <Text>
          <![CDATA[ NOP(); ]]>
        </Text>
      </Rung>
    - <Rung Number="1" Type="N">
      - <Comment>
          <![CDATA[ Get the Controllers real time clock broken down
          by Year, month, day...microseconds and store in a tag so that the values can
        </Comment>
      - <Text>
          <![CDATA[ XIC(Test_CurrentDate)GSV
          (WALLCLOCKTIME,,DateTime,TestDateTime.Year); ]]>
        </Text>
      </Rung>
    - <Rung Number="2" Type="N">
      - <Text>
          <![CDATA[ MOV(TestDateTime.Year,TestDateTime.Year)MOV
          (TestDateTime.Month,TestDateTime.Month)MOV(TestDateTime.Day,TestDateTime.Day)
        </Text>
      </Rung>
    - <Rung Number="3" Type="N">
      - <Text>
          <![CDATA[ JSR(DayOfWeek,1,TestDateTime,Test_Result)MOV
          (Test_Result,Test_Result); ]]>
        </Text>
      </Rung>
  </RLLContent>
</Routine>
```

## L5K examples

**Ladder ROUTINE example**




```
ROUTINE Ladder_example
       RC: "This is a rung comment for the first rung.";
       N: XIC(input1)XIC(input2)OTE(output1)OTE(output2);
       RC: "This is a rung comment for the second rung.";
       N: XIC(input3)OTE(output3);
END_ROUTINE
```

**Example with a single branch**

```
N: XIC(conveyor_a)[,XIC(input_1) XIO(input_2) ]OTE(light_1);
```

**Example with two simultaneous branches**

```
N: XIC(conveyor_b)[,XIC(input_1) XIO(input_2),XIC(input_a)
XIO(input_b)  ]OTE(light_2);
```

## Neutral text for ladder instructions

These tables lists each ladder instruction and its neutral text format. For details about a specific instruction, see the listed reference manuals.

**Ladder Logic Instruction Reference Manuals**

| Instruction Type | Resource |
|---|---|
| Basic, sequential instruction | Logix5000 Controllers General Instructions Set Reference Manual, publication 1756-RM003. |
| Process control or drives instruction | Logix5000 Controllers Advanced Process Control and Drives Instruction Set Reference Manual, publication 1756-RM006. |
| Motion instruction | Logix5000 Controllers Motion Instructions Reference Manual, publication MOTION-RM002. |

**Relay Ladder Instructions**

| Instruction | Neutral Text Format |
|---|---|
| ABL | ABL(*channel,serial_port_control,character_count*); |
| ABS | ABS(*source,destination*); |
| ACB | ACB(*channel,serial_port_control,character_count*); |
| ACL | ACL(*channel,clear_serial_port_read,clear_serial_port_write*); |
| ACS | ACS(*source,destination*); |
| ADD | ADD(*source_A,source_B,destination*); |
| AFI | AFI(); |
| AHL | AHL(*channel,ANDMask,ORMask,serial_port_control,channel_status*); |
| ALMA | ALMA (*alma_tag,in,program_acknowledge_all,program_disable,program_enable*); |
| ALMD | ALMD (*almd_tag,program_acknowledge,program_reset,program_disable, program_enable*); |
| AND | AND(*source_A,source_B,destination*); |
| ARD | ARD(*channel,destination,serial_port_control,string_length, characters_read*); |
| ARL | ARL(*channel,destination,serial_port_control,string_length, characters_read*); |
| ASN | ASN(*source,destination*); |
| ATN | ATN(*source,destination*); |
| AVC | AVC(*avc_tag,feedback_type,feedback_reation_time,delay_type,delay_time, output_follows_actuate,actuate,delay_enable,feedback_1,input_status, output_status,reset*); |
| AVE | AVE(*array,dim_to_vary,destination,control,length,position*); |
| AWA | AWA(*channel,source,serial_port_control,string_length,characters_sent*); |
| AWT | AWT(*channel,source,serial_port_control,string_length,characters_sent*); |
| BRK | BRK(); |
| BSL | BSL(*array,control,source_bit,length*); |

| Instruction | Neutral Text Format |
|---|---|
| BSR | BSR(*array,control,source_bit,length*); |
| BTD | BTD(*source,source_bit,destination,destination_bit,length*); |
| CBCM | CBCM(*cbcm_tag,ack_type,mode,takeover_mode,enable,safety_enable, standard_enable,arm_continuous,start,stop_at_top,press_in_motion, motion_monitor_fault,slide_zone,safety_enable_ack*) |
| CBIM | CBIM(*cbim_tag,ack_type,inch_time,enable,safety_enable,standard_enable, start,press_in_motion,motion_monitor_fault,slide_zone,safety_enable_ack*); |
| CBSSM | CBSSM(*cbssm_tag,ack_type,takeover_mode,enable,safety_enable, standard_enable,start,press_in_motion,motion_monitor_fault,slide_zone, saefty_enable_ack*); |
| CLR | CLR(*destination*); |
| CMP | CMP(*expression*); |
| CONCAT | CONCAT(*sourceA,sourceB,destination*) |
| COP | COP(*source,destination,length*); |
| COS | COS(*source,destination*); |
| CPM | CPM(*cpm_tag,cam_profile,enable,brake_cam,takeover_cam,dynamic_cam, input_status,reverse,press_motion_status,reset*); |
| CPS | CPS(*source,destination,length*); |
| CPT | CPT(*destination,expression*); |
| CROUT | CROUT(*crout_tag,feedback_type,feedback_reaction_time,actuate,feedback_1, feedback_2,input_status,output_status,reset*); |
| CSM | CSM(*csm_tag,mechanical_delay_timer,max_pulse_period,motion_request, channel_A,channel_B,input_status,reset*); |
| CTD | CTD(*counter,preset,accum*); |
| CTU | CTU(*counter,preset,accum*); |
| DCM | DCM(*dcm_tag,safety_function,input_type,descrepancy_time,channel_A, channel_B,input_status,reset*); |
| DCS | DCS(*dcs_tag,safety_function,input_type,discrepancy_time,restart_type, cold_start_type,channel_A,channel_B,input_status,reset*); |
| DCSRT | DCSRT(*dcsrt_tag,safety_function,input_type,discrepancy_time,enable, channel_A,channel_B,input_status,reset*); |
| DCST | DCST(*dcst_tag,safety_function,input_type,discrepancy_time,restart_type, cold_start_type,channel_A,channel_B,test_request,input_status,reset*); |
| DCSTM | DCSTM(*dcstm_tag,safety_function,input_type,discrepancy_time,restart_type, cold_start_type,test_type,test_time,channel_A,channel_B,test_request,mute, muting_lamp_status,input_status,reset*); |
| DCSTL | DCSTL(*dcstl_tag,safety_function,input_type,discrepancy_time,restart_type, cold_start_type,channel_A,channel_B,test_request,unlock_request, lock_feedback,hazard_stopped,input_status,reset*); |
| DDT | DDT(*source,reference,result,cmp_control,length,position,result_control, length,position*); |
| DEG | DEG(*source,destination*); |
| DELETE | DELETE(*source,quantity,start,destination*); |
| DIN | DIN(*din_tag,reset_type,channel_A,channel_B,circuit_reset,fault_reset*); |
| DIV | DIV(*source_A,source_B,destination*); |

| Instruction | Neutral Text Format |
|---|---|
| DTOS | DTOS(*source,destination*); |
| DTR | DTR(*source,mask,reference*); |
| ENPEN | ENPEN(*enpen_tag,reset_type,channel_A,channel_B,circuit_reset,fault_reset*); |
| EOT | EOT(*data_bit*); |
| EPMS | EPMS(*epms_tag,input_1,input_2,input_3,input_4,input_5,input_6,input_7, input_8,input_status,lock,reset*); |
| EQU | EQU(*source_A,source_B*); |
| ESTOP | ESTOP(*estop_tag,reset_type,channel_A,channel_B,circuit_reset,fault_reset*); |
| EVENT | EVENT(*task*); |
| FAL | FAL(*control,length,position,mode,destination,expression*); |
| FBC | FBC(*source,reference,result,cmp_control,length,position,result_control, length,position*); |
| FFL | FFL(*source,FIFO,control,length,position*); |
| FFU | FFU(*FIFO,destination,control,length,position*); |
| FIND | FIND(*source,search,start,result*); |
| FLL | FLL(*source,destination,length*); |
| FOR | FOR(*routine_name,index,initial_value,terminal_value,step_size*); |
| FPMS | FPMS(*fpms_tag,input_1,input_2,input_3,input_4,input_5,fault_reset*); |
| FRD | FRD(*source,destination*); |
| FSBM | FSBM(*fsbm_tag,restart_type,S1-S2_time,S2-LC_time,LC-S3_time,S3-S4_time, maximum_mute_time,maximum_override_time,direction,light_curtain,sensor_1, sensor_2,sensor_3,sensor_4,enable_mute,override,input_status, muting_lamp_status,reset*); |
| FSC | FSC(*control,length,position,mode,expression*); |
| GEQ | GEQ(*source_A,source_B*); |
| GRT | GRT(*source_A,source_B*); |
| GSV | GSV(*class_name,instance_name,attribute_name,destination*); |
| INSERT | INSERT(*sourceA,sourceB,start,destination*); |
| IOT | IOT(*output_tag*); |
| JMP | JMP(*label_name*); |
| JSR | JSR(*routine_name,input_1,...input_n,return_1,..return_n*); |
| JXR | JXR(*external_routine_name,external_routine_control,parameter, return_parameter*); |
| LBL | LBL(*label_name*); |
| LC | LC(*lc_tag,reset_type,channel_A,channel_B,input_filter_time, mute_light_curtain,circuit_reset,fault_reset*); |
| LEQ | LEQ(*source_A,source_B*); |
| LES | LES(*source_A,source_B*); |
| LFL | LFL(*source,LIFO,control,length,position*); |
| LFU | LFU(*LIFO,destination,control,length,position*); |
| LIM | LIM(*low_limit,test,high_limit*); |
| LN | LN(*source,destination*); |

| Instruction | Neutral Text Format |
|---|---|
| LOG | LOG(*source,destination*); |
| LOWER | LOWER(*source,destination);* |
| MAAT | MAAT(*axis,motion_control*); |
| MAFR | MAFR(*axis,motion_control*); |
| MAG | MAG(slave*_axis,master_axis,motion_control,direction,ratio,slave_counts,master_counts,master_reference,ratio_format,clutch,accel_rate,accel_ units*); |
| MAH | MAH(*axis,motion_control*); |
| MAHD | MAHD(*axis,motion_control,diagnostic_test,observed_direction*); |
| MAJ | MAJ(*axis,motion_control,direction,speed,speed_units,accel_rate, accel_units,decel_rate,decel_units,profile,merge,merge_speed*); |
| MAM | MAM(*axis,motion_control,move_type,position,speed,speed_units,accel_rate, accel_units,decel_rate,decel_units,profile,merge,merge_speed*); |
| MAOC | MAOC(*axis,execution_target,motion_control,output,input,output_cam, cam_start_position,cam_end_position,output_compensation,execution_mode, execution_schedule,axis_arm_position,cam_arm_position,reference*); |
| MAPC | MAPC(*slave_axis,master_axis,motion_control,direction,cam_profile, slave_scaling,master_scaling,execution_mode,execution_schedule, master_lock_position,cam_lock_position,master_reference, master_direction*); |
| MAR | MAR(*axis,motion_control,trigger_condition,windowed_registration, minimum_position,maximum_position*); |
| MAS | MAS(*axis,motion_control,stop_type,change_decel,decel_rate,decel_units*); |
| MASD | MASD(*axis,motion_control*); |
| MASR | MASR(*axis,motion_control*); |
| MATC | MATC(*axis,motion_control,direction,cam_profile,distance_scaling, time_scaling,execution_mode,execution_schedule*); |
| MAW | MAW(*axis,motion_control,trigger_condition,position*); |
| MCCD | MCCD(*coordinate_system,motion_control,motion_type,change_speed,speed, speed_units,change_accel,accel_rate,accel_units,change_decel,decel_rate, decel_units,scope*); |
| MCCM | MCCM(*coordinate_system,motion_control,move_type,position,circle_type, via/center/radius,direction,speed,speed_units,accel_rate,accel_units, decel_rate,decel_units,profile,termination_type,merge,merge_speed*); |
| MCCP | MCCP(*motion_control,cam,length,start_slope,end_slope,cam_profile*); |
| MCLM | MCLM(*coordinate_system,motion_control,move_type,position,speed, speed_units,accel_rate,accel_units,decel_rate,decel_units,profile, termination_type,merge,merge_speed*); |
| MCD | MCD(a*xis,motion_control,motion_type,change_speed,speed,change_accel, accel_rate,change_decel,decel_rate,speed_units,accel_units, decel_units*); |
| MCR | MCR(); |
| MCS | MCS(*coordinate_system,motion_control,stop_type,change_decel,decel_rate, decel_units*); |
| MCSD | MCSD(*coordinate_system,motion_control*); |
| MCSR | MCSR(*coordinate_system,motion_control*); |
| MCSV | MCSV(*motion_control,cam_profile,master_value,slave_value,slope_value, slope_derivative*); |
| MCT | MCT(*source_system,target_system,motion_control,orientation,translation*); |

| Instruction | Neutral Text Format |
|---|---|
| MCTP | MCTP(*source_system,target_system,motion_control,orientation,translation, transform_direction,reference_position,transform_position*); |
| MDF | MDF(*axis,motion_control*); |
| MDO | MDO(*axis,motion_control,drive_output,drive_units*); |
| MDOC | MDOC(*axis,execution_target,motion_control,disarm_type*); |
| MDR | MDR(*axis,motion_control*); |
| MDW | MDW(*axis,motion_control*); |
| MEQ | MEQ(*source,mask,compare*); |
| MGS | MGS(*group,motion_control,stop_mode*); |
| MGSD | MGSD(*group,motion_control*); |
| MGSP | MGSP(*group,motion_control*); |
| MGSR | MGSR(*group,motion_control*); |
| MID | MID(*source,quantity,start,destination*); |
| MMVC | MMVC(*mmvc_tag,enable,keyswitch,bottom,flywheel_stopped,safety_enable, actuate,input_status,output_status,reset*); |
| MOD | MOD(*source_A,source_B,destination*); |
| MOV | MOV(*source,destination*); |
| MRAT | MRAT(*axis,motion_control*); |
| MRHD | MRHD(*axis,motion_control,diagnostic_test*); |
| MRP | MRP(a*xis,motion_control,type,position_select,position*); |
| MSF | MSF(*axis,motion_control*); |
| MSG | MSG(*message_control*); |
| MSO | MSO(*axis,motion_control*); |
| MUL | MUL(*source_A,source_B,destination*); |
| MVC | MVC(*mvc_tag,feedback_type,feedback_reaction_time,actuate,feedback_1, feedback_2,input_status,output_status,reset*); |
| MVM | MVM(*source,mask,destination*); |
| NEG | NEG(*source,destination*); |
| NEQ | NEQ(*source_A,source_B*); |
| NOP | NOP(); |
| NOT | NOT(*source,destination*); |
| ONS | ONS(*storage_bit*); |
| OR | OR(*source_A,source_B,destination*); |
| OSF | OSF(*storage_bit,output_bit*); |
| OSR | OSR(*storage_bit,output_bit*); |
| OTE | OTE(*data_bit*); |
| OTL | OTL(*data_bit*); |
| OTU | OTU(*data_bit*); |
| PATT | PATT(*phase_name,result*); |
| PCLF | PCLF(*phase_name*); |
| PCMD | PCMD(*phase_name,command,result*); |

| Instruction | Neutral Text Format |
|---|---|
| PDET | PDET(*phase_name*); |
| PFL | PFL(*source*); |
| PID | PID(PID,*process_variable,tieback,control_variable,pid_master_loop,*<br>*inhold_bit,inhold_value);* |
| POVR | POVR(*phase_name,command,result);* |
| PPD | PPD(); |
| PRNP | PRNP(); |
| PSC | PSC(); |
| PXRQ | PXRQ(*phase_instruction,external_request,data_value*); |
| RAD | RAD(*source,destination*); |
| RES | RES(*structure*); |
| RET | RET(*return_1,...return_n*); |
| RIN | RIN(*rin_tag,reset_type,channel_A,channel_B,circuit_reset,fault_reset*); |
| ROUT | ROUT(*rout_tag,feedback_type,enable,feedback_1,feedback_2,fault_reset*); |
| RTO | RTO(*timer,preset,accum*); |
| RTOS | RTOS(*source,destination*) |
| SBR | SBR(*routine_name,input_1,...input_n*); |
| SFP | SFP(SFC_routine_name,target_state); |
| SFR | SFR(SFC_routine_name,step_name); |
| SIN | SIN(*source,destination*); |
| SIZE | SIZE(*souce,dimension_to_vary,size*); |
| SMAT | SMAT(*smat_tag,restart_type,short_circuit_detect_delay_time,channel_A,*<br>*channel_B,input_status,reset*); |
| SQI | SQI(*array,mask,source,control,length,position*); |
| SQL | SQL(*array,source,control,length,position*); |
| SQO | SQO(*array,mask,destination,control,length,position*); |
| SQR | SQR(*source,destination*); |
| SRT | SRT(*array,dim_to_vary,control,length,position*); |
| SSV | SSV(*class_name,instance_name,attribute_name,source*); |
| STD | STD(*array,dim_to_vary,destination,control,length,position*); |
| STOD | STOD(*source,destination*) |
| STOR | STOR(*source,destination*) |
| SUB | SUB(*source_A,source_B,destination*); |
| SWPB | SWPB(*source,order_mode,destination*); |
| TAN | TAN(*source,destination*); |
| THRS | THRS(*thrs_tag,active_pin_type,active_pin,right_button_normally_open,*<br>*right_button_normally_closed,left_button_normally_open,*<br>*left_button_normally_closed,fault_reset*); |
| THRSE | THRSE(*thrse_tag,discprepancy_time,enable,disconnected,*<br>*right_button_normally_open,right_button_normally_closed,*<br>*left_button_normally_open,left_button_normally_closed,input_status,resest*); |
| TND | TND(); |

| Instruction | Neutral Text Format |
|---|---|
| TOD | TOD(*source,destination*); |
| TOF | TOF(*timer,preset,accum*); |
| TON | TON(*timer,preset,accum*); |
| TRN | TRN(*source,destination*); |
| TSAM | TSAM(*tsam_tag,restart_type,S1-S2_time,S2-LC_time,maximum_mute_time, maximum_override_time,light_curtain,sensor_1,sensor_2,enable_mute, override,input_status,muting_lamp_status,reset*); |
| TSSM | TSSM(*tssm_tag,restart_type,S1-S2_discrepancy_time,S1_S2-LC_minimum_time, S1_S2-LC_maximum_time,maximum_mute_time,maximum_override_time, light_curtain,sensor_1,sensor_2,enable_mute,override,input_status, muting_lamp_status,reset*); |
| UID | UID(); |
| UIE | UIE(); |
| UPPER | UPPER(*source,destination);* |
| XIC | XIC(*data_bit*); |
| XIO | XIO(*data_bit*); |
| XOR | XOR(*source_A,source_B,destination*); |
| XPY | XPY(*source_A,source_B,destination*); |

# Define a function block diagram routine

## Introduction

This chapter explains how to enter function block diagram logic in a complete import/export file.

## Function BlockDiagram Routine

These examples show the function block routine structure.

### L5X function block diagram routine structure

```
<Routines>
      <Routine [Routine_Attributes]>
            <Description>
                  <![CDATA[ text ]]>
            </Description>
            <FBDContent [FBDContent_Attributes]>
                  <Sheet [Sheet_Attributes]>
                        logic
                  </Sheet>
            </FBDContent>
      </Routine>
</Routines>
```

### L5K Function Block FBD_ROUTINE structure

```
FBD_ROUTINE <routine_name> [(Description := "text",
      Routine_attributes, FBD_Attributes)]
      <function block sheets>
END_FBD_ROUTINE
```

### Function block routine elements

| L5X Item | L5K Item | Identifies |
|---|---|---|
| N/A | *routine_name* | The name of the routine.<br><br>In L5X, use a Name attribute on the <Routine> element. |
| Description | Description | User information about the routine. |
| FBDContent | *function block sheets* | Function block diagram logic. |
| EncryptionInfo | ENCRYPTION_INFO | Details of the license-based Source Protection for the lockable object. Only exists for protected routines exported in plain text. |
| EncryptedContent | N/A | Source Protected and locked routine content. Only exists for locked routines exported in plain text. |
| EncryptedSegments | N/A | Locked logic for the routine. Only exists for locked routines exported in plain text. |

## FBD_routine attributes

| Attribute | Description |
|---|---|
| Name | L5X only. Specify the name of the routine.<br><br>In L5K, the name is an element of the statement. |
| Type | L5X only. Specify FBD.<br><br>In L5K, the type of routine is part of the routine statement. |
| SheetSize | Select one of these sizes.<br>• Letter (8.5x11in)<br>• Legal (8.5x14in)<br>• Tabloid (11x17in)<br>• A4 (210x297mm)<br>• A3 (297x420mm)<br><br>For L5X, this attribute is on the <FBDContent> element. |
| SheetOrientation | Select the orientation of the sheet as Portrait or Landscape.<br><br>For L5X, this attribute is on the <FBDContent> element. |
| OnlineEditType | L5X only. Specify the online edit logic type (**Original**, **PendingEdits**, or **TestEdits**). This attribute is not specified if there are no edits.<br><br>For L5X, this attribute is on the <FBDContent> element. |
| PermissionSet | Name of the set of permissions, configured in FactoryTalk Security, to apply to this object. |
| TrackingGroups | The group of tracked objects to which this item belongs. Components can be marked for tracking to determine whether they have been changed. Version 30 of the Logix Designer application supports only one tracking group. |

## Sheet

Enter the function block diagram logic in sheets within a routine component in an import/export file.

### L5X sheet structure

```
<Sheet [Sheet_Attributes]>
       <Description>
            <![CDATA[ text ]]>
       </Description>
      <IRef [Iref_Attributes]/>
      <ORef [Oref_Attributes]/>
      <ICon [Icon_Attributes]/>
      <OCon [Ocon_Attributes]/>
      <Block [Block_Attributes]/>
      <AddOnInstruction [AddOnInstruction_Attributes]/>
      <JSR [JSR_Attributes]/>
      <SBR [SBR_Attributes]/>
```

```
        <RET [RET_Attributes]/>
        <Wire [Wire_Attributes]/>
        <FeedbackWire [FeedbackWire_Attributes]/>
        <Function [Function_Attributes]/>
        <TextBox [TextBox_Attributes]>
              text
        </TextBox>
        <Attachment [Attachment_Attributes]/>
</Sheet>
```

## L5K SHEET structure

```
SHEET (Name := <sheet_name>)
        <IREF declaration>
        <OREF declaration>
        <ICON declaration>
        <OCON declaration>
        <mnemonic_BLOCK declaration>
        <ADD_ON_INSTRUCTION declaration>
        <JSR declaration>
        <SBR declaration>
        <RET declaration>
        <WIRE declaration>
        <FEEDBACK_WIRE declaration>
        <FUNCTION declaration>
        <TEXT_BOX declaration>
        <ATTACHMENT declaration>
END_SHEET
```

## Sheet elements

| L5X Item | L5K Item | Identifies |
|---|---|---|
| IRef | IREF | Input references. |
| ORef | OREF | Output references. |
| ICon | ICON | Input wire connectors. |
| OCon | OCON | Output wire connectors. |
| Block | mnemonic_BLOCK | Function block instructions and their locations. |
| AddOnInstruction | ADD_ON_INSTRUCTION | Add-On Instructions. |
| JSR | JSR | Jump to Subroutine instructions. |
| SBR | SBR | Subroutine instructions. |
| RET | RET | Return instructions. |
| Wire | WIRE | Wires and their corresponding attachments. |
| FeedbackWire | FEEDBACK_WIRE | Feedback wires and what they are attached to. |
| Function | FUNCTION | Function block functions. |
| TextBox | TEXT_BOX | Text box to hold comments. |
| Attachment | ATTACHMENT | Attachment from a text box to another function block element. |

## Sheet attributes

| Attribute | Description |
|---|---|
| Number | L5X only. Specify the number of the sheet.<br><br>In L5K, the sheet number is determined by the order the SHEET statement appears in the L5K format. |
| Name | L5K only. Specify the name of the sheet.<br><br>In L5X, the name of the sheet is specified in a **<Description>** element under the **<Sheet>** element. |

## Sheet guidelines

Use these sheet guidelines in your import/export files:

- The sheets in the routine appear in order in the export file. Each sheet section contains all the drawing elements and wires for that sheet.
- On import, sheet numbers are assigned based on order in the file, not on the number attribute on the sheet.
- The sheet name is stored as description on the sheet.
- Input references, blocks, output references, special drawing elements, and wires are contained within the sheet. On export, the elements appear in the order shown. On import in the L5K format, elements can be interspersed in the file. On import in the L5X format, the elements must appear in the exported order.
- Wire and feedback wire statements must appear after all the other components.
- Be careful when copying and pasting function block components within an import/export file. Each component within a sheet must have a unique ID number within that sheet.

## Export function block logic while editing online

If you export function block logic that contains online edits, the export file exports LOGIC blocks, in L5K format, or additional <FBDContent> elements, in L5X format, to indicate the original, test edits, and pending edits states. If there are no online edits, these LOGIC blocks or additional <FBDContent> elements are not shown.

#### L5X Example: Test Edits and Pending Edits Exist

```
- <Routine Name="FBDRoutine" Type="FBD">
  - <FBDContent SheetSize="Letter - 8.5 x 11 in" SheetOrientation="Landscape"
    OnlineEditType="Original">
      <!--  Sheets inserted here - see format descibed above  -->
    </FBDContent>
  - <FBDContent SheetSize="Letter - 8.5 x 11 in" SheetOrientation="Landscape"
    OnlineEditType="PendingEdits">
      <!--  Sheets inserted here - see format descibed above  -->
    </FBDContent>
  - <FBDContent SheetSize="Letter - 8.5 x 11 in" SheetOrientation="Landscape"
    OnlineEditType="TestEdits">
      <!--  Sheets inserted here - see format descibed above  -->
    </FBDContent>
  </Routine>
```

#### L5K Example 1: Both Test Edits and Pending Edits Exist

```
FBD_ROUTINE MyFbdRoutine (SheetSize := "Letter (8.5x11in)",
SheetOrientation := Landscape)
       LOGIC (Online_Edit_Type := Orig)
              (* Sheets inserted here - see format described
above *)
       END_LOGIC

       LOGIC (Online_Edit_Type := Test)
              (* Sheets inserted here - see format described
above *)
       END_LOGIC

       LOGIC (Online_Edit_Type := Pend)
              (* Sheets inserted here - see format described
above *)
       END_LOGIC
END_FBD_ROUTINE
```

#### L5K Example 2: Only Pending Edits Exist

```
FBD_ROUTINE MyFbdRoutine (SheetSize := "Letter (8.5x11in)",
SheetOrientation := Landscape)
       LOGIC (Online_Edit_Type := Orig)
              (* Sheets inserted here - see format described
above *)
       END_LOGIC

       LOGIC (Online_Edit_Type := Pend)
              (* Sheets inserted here - see format described
above *)
       END_LOGIC
END_FBD_ROUTINE
```

**Online Edit Types**

| Item | Identifies |
|---|---|
| Online_Edit_Type | If online edits exist when the logic is exported, there will be a LOGIC block for Online_Edit_Type := Orig and the appropriate LOGIC block for the existing edits. Online_Edit_Type : = Pend indicates pending edits. Online_Edit_Type := Test indicates test edits.<br>If there are no online edits when the logic is exported, there are no LOGIC blocks and the main components in the routine are SHEET components. |

# Input and output references

Input and output references have similar formats and identical attributes.

## L5X IREFand OREF structure

```
<IRef [Reference_Attributes]/>

<ORef [Reference_Attributes]/>
```

## L5K IREF and OREF structure

```
IREF (Reference_Attributes)
END_IREF

OREF (Reference_Attributes)
END_OREF
```

## Reference attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| ID | ID | The element identifier; uniqueness is important for wiring.<br>Type an unsigned, 32-bit integer value. |
| X | X | X-coordinates on internal grid.<br>Type an unsigned, 32-bit integer value. |
| Y | Y | Y-coordinates on internal grid.<br>Type an unsigned, 32-bit integer value. |
| Operand | Operand | Function block instruction. |
| HideDesc | HideDescription | Whether or not to hide the description. Specify **true** or **false**. |

## Reference guidelines

Use these reference guidelines for function blocks:

- If the operand is not a qualified tag or literal value, the reference is not verified.
- The X and Y grid locations are a relative position from the upper-left corner of the sheet. X is the horizontal position; Y is the vertical position.

## Input and output connectors

Input and output wire connectors have similar formats and identical attributes.

### L5X ICON and OCON structure

```
<ICon [Connector_Attributes]/>

<OCon [Connector_Attributes]/>
```

### L5K ICON and OCON structure

```
ICON (Connector_Attributes)
END_ICON

OCON (Connector_Attributes)
END_OCON
```

### Connector attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| ID | ID | The element identifier; uniqueness is important for wiring.<br>Type an unsigned, 32-bit integer value. |
| X | X | X-coordinates on internal grid.<br>Type an unsigned, 32-bit integer value. |
| Y | Y | Y-coordinates on internal grid.<br>Type an unsigned, 32-bit integer value. |
| Name | Name | (optional) The name of the wire connector. |

### Connector guidelines

Use these connector guidelines for function blocks:

- Output connector names must be unique within a function block routine.
- Multiple input connector names can reference the same output connector name.
- Input and output connectors with unmatched or blank connector names are not verified.
- The X and Y grid locations are a relative position from the upper-left corner of the sheet. X is the horizontal position; Y is the vertical position.

## Blocks

These examples show block structure.

### L5X block structure

```
<Block [Block_Attributes]>
      <Array Name="name", Operand="operand"/>
</Block>
```

### L5K BLOCK structure

```
mnemonic_BLOCK (Block_Attributes)
END_mnemonic_BLOCK
```

## Block attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| Type | *mnemonic* | L5X only. The type of Block. Specify the mnemonic name for the block (for example *DEDT*). |
| ID | ID | The element identifier; uniqueness is important for wiring.<br>Type an unsigned, 32-bit integer value. |
| X | X | X-coordinates on internal grid.<br>Type an unsigned, 32-bit integer value. |
| Y | Y | Y-coordinates on internal grid.<br>Type an unsigned, 32-bit integer value. |
| Operand | Operand | (optional) Tag name for the block.<br>For L5X, this attribute is on the <Array> element. |
| Name | ArrayName | (optional) Tag name for array.<br>For L5X, this attribute is on the <Array> element. |
| VisiblePins | VisiblePins | List of the names of all the parameters with pins visible for wiring. The names match the member names on the block tag.<br>In an.L5X file, separate pin names with spaces.<br>In an.L5K file, separate pin names with commas. |
| AutotuneTag | AutotuneTag | Tag name for the autotune tag. |
| HideDesc | HideDescription | Whether or not to hide the description. Specify true or false. |

## Block guidelines

Use these block guidelines with function blocks:

- If the operand is not a qualified tag of the correct data type, the blocks are not verified.
- Some function block instructions require specific arrays. The table lists the valid array name for each of these instructions.

| Instruction | Array Name |
|---|---|
| DEDT | Storage (required) |
| FGEN | X1 (required)<br>Y1 (required)<br>X2 (optional)<br>Y2 (optional) |
| MAVE | Storage (required)<br>Weight (optional) |
| RMPS | RampValue (required)<br>SoakValue (required)<br>SoakTime (required) |

- The X and Y grid locations are in a relative position from the upper-left corner of the sheet. X is the horizontal position; Y is the vertical position.

## Functions

Function Block functions are similar to instructions but do not require backing tags, require less memory, sometimes execute more quickly, and take up less space in a function block diagram. Function Block functions are available only on CompactLogix 5380, Compact GuardLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

### L5X function structure

```
<Function [Function_Attributes]/>
```

### L5K FUNCTION structure

```
FUNCTION (Function_Attributes)
END_FUNCTION
```

### Function attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| Type | *mnemonic* | L5X only. The type of function. For L5K, specify the mnemonic name for the function (for example, *ADD*). |
| ID | ID | The element identifier; uniqueness is important for wiring.<br>Type an unsigned, 32-bit integer value. |
| X | X | X-coordinates on internal grid.<br>Type an unsigned, 32-bit integer value. |
| Y | Y | Y-coordinates on internal grid.<br>Type an unsigned, 32-bit integer value. |

## Add-On instructions

The Add-On Instruction elements in a function block diagram routine structure represent the use of the Add-On Instruction in the routine, not the definition of the Add-On Instruction.

### L5X Add-On Instruction structure

<AddOnInstruction [*AddOnInstruction_Attributes*]>

<InOutParameter Name="*InOutArgument*", Argument="*argument*"/>

</Add0nInstruction>

### L5K ADD_ON_INSTRUCTION structure

```
ADD_ON_INSTRUCTION name (Add_On_Instruction_Attributes)
       FBD_PARAMETERS (InOutParmName := InOutArgument,...)
       END_FBD_PARAMETERS
END_ADD_ON_INSTRUCTION
```

### Add-On Instruction Attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| Name | *name* | L5X only. Specify the name of the Add-On Instruction.<br><br>In L5K, the name is an element of the statement. |

| L5X Item | L5K Item | Identifies |
|---|---|---|
| ID | ID | The element identifier; uniqueness is important for wiring.<br>Type an unsigned, 32-bit integer value. |
| X | X | X-coordinates on internal grid.<br>Type an unsigned, 32-bit integer value. |
| Y | Y | Y-coordinates on internal grid.<br>Type an unsigned, 32-bit integer value. |
| Operand | Operand | (optional) Operand name for the Add-On Instruction block. |
| VisiblePins | VisiblePins | List of the names of all the parameters with pins visible for wiring.<br>In an.L5X file, separate pin names with spaces.<br>In an.L5K file, separate pin names with commas. |
| Name | N/A | L5X only. Specify the tag name for the InOut Parameter on the Add-On Instruction<br><br>For L5X, this attribute is on the <InOutParameter> element.<br>For L5K, the InOut Parameter name is one of the attribute names on the FBD_PARAMETERS statement. |
| Argument | N/A | L5X only. Specify the InOut Parameter argument.<br><br>For L5X, this attribute is on the <InOutParameter> element. |

## Add-On Instruction Guidelines

Use these Add-On Instruction guidelines with function blocks:

- If the operand is not a qualified tag or literal value, the Add-On Instruction is not verified.
- The X and Y grid locations are a relative position from the upper-left corner of the sheet. X is the horizontal position; Y is the vertical position.

# JSR

## L5X JSR structure

<JSR [*JSR_Attributes*]>

</JSR>

## L5K JSR structure

```
JSR (JSR_Attributes)
END_JSR
```

## JSR attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| ID | ID | The element identifier; uniqueness is important for wiring.<br>Type an unsigned, 32-bit integer value. |
| X | X | X-coordinates on internal grid.<br>Type an unsigned, 32-bit integer value. |
| Y | Y | Y-coordinates on internal grid.<br>Type an unsigned, 32-bit integer value. |

| L5X Item | L5K Item | Identifies |
|---|---|---|
| Routine | Routine | Specify the JSR routine name. |
| In | In | Specify the input parameters.<br><br>In an.L5X file, separate input parameter names with spaces.<br>In an.L5K file, separate input parameter names with commas. |
| Ret | Ret | Specify the return parameters.<br><br>In an.L5X file, separate return parameter names with spaces.<br>In an.L5K file, separate return parameter names with commas. |

### JSR guidelines

The X and Y grid locations are a relative position from the upper-left corner of the sheet. X is the horizontal position; Y is the vertical position.

## SBR

### L5X SBR structure

```
<SBR [SBR_Attributes]>
</SBR>
```

### L5K SBR structure

```
SBR (SBR_Attributes)
END_SBR
```

### SBR attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| ID | ID | The element identifier; uniqueness is important for wiring.<br>Type an unsigned, 32-bit integer value. |
| X | X | X-coordinates on internal grid.<br>Type an unsigned, 32-bit integer value. |
| Y | Y | Y-coordinates on internal grid.<br>Type an unsigned, 32-bit integer value. |
| Routine | Routine | Specify the SBR routine name. |
| Ret | Ret | Specify the return parameters.<br><br>In an.L5X file, separate return parameter names with spaces.<br>In an.L5K file, separate return parameter names with commas. |

### SBR guidelines

The X and Y grid locations are a relative position from the upper-left corner of the sheet. X is the horizontal position; Y is the vertical position.

# RET

## L5X RET structure

```
<RET [RET_Attributes]>
</RET>
```

## L5K RET structure

```
RET (RET_Attributes)
END_RET
```

## RET attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| ID | ID | The element identifier; uniqueness is important for wiring.<br>Type an unsigned, 32-bit integer value. |
| X | X | X-coordinates on internal grid.<br>Type an unsigned, 32-bit integer value. |
| Y | Y | Y-coordinates on internal grid.<br>Type an unsigned, 32-bit integer value. |
| Routine | Routine | Specify the JSR routine name. |
| In | In | Specify the input parameters.<br><br>In an.L5X file, separate input parameter names with spaces.<br>In an.L5K file, separate input parameter names with commas. |

## RET guidelines

The X and Y grid locations are a relative position from the upper-left corner of the sheet. X is the horizontal position; Y is the vertical position.

# Wires and feedback wires

The wire and feedback wire formats describe a wire by specifying what it is attached to at each end, which is always a pin on another drawing element.

## L5X wire structure

```
<Wire [Wire_Attributes]/>

<FeedbackWire [Wire_Attributes]/>
```

## L5K WIRE structure

```
WIRE (Wire_Attributes)
END_WIRE

FEEDBAK_WIRE (Wire_Attributes)
END_FEEDBACK_WIRE
```

## Wire attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| FromID | FromElementID | The source drawing element.<br>Type an unsigned, 32-bit integer. |

| L5X Item | L5K Item | Identifies |
|---|---|---|
| FromParam | FromParameter | The pin on the source drawing element.<br><br>**For:** **Type:**<br>Blocks Parameter name<br>Irefs In<br>Icons In |
| ToID | ToElementID | The destination drawing element.<br>Type an unsigned, 32-bit integer. |
| ToParam | ToParameter | The pin on the destination drawing element.<br><br>**For:** **Type:**<br>Blocks Parameter name<br>Orefs Out<br>Ocons Out |

## Wire guidelines

Use these wire guidelines:

- Wires that are not correctly specified are not imported.
- A feedback wire follows the same format as a wire. Just connect the Source and Destination elements to form a feedback wire.

# Text boxes

The text box blocks in an SFC routine hold descriptions about SFC components.

## L5X TextBox structure

```
<TextBox [TextBoxAttributes]>
      <![CDATA[ text  ]] >
</TextBox>
```

## L5K TEXTBOX structure

```
TEXT_BOX (Text_Box_Attributes,
      Text := <"text">)
END_TEXT_BOX
```

## Text box attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| ID | ID | The element identifier; uniqueness is important for wiring.<br>Type an unsigned, 32-bit integer value. |
| X | X | X-coordinates on internal grid.<br>Type an unsigned, 32-bit integer value. |
| Y | Y | Y-coordinates on internal grid.<br>Type an unsigned, 32-bit integer value. |
| Width | Width | This attribute is not currently used; it is there for future use. Type **0**. |
| Text | Text | The descriptive text. |

### Text box guidelines

Use these guidelines for text boxes:

- All text box blocks must come after all block sections.
- Text boxes can be free-standing or they can be attached to FBD elements.

## Attachments

The attachment blocks identify the attachments from text boxes to other function block elements.

### L5X attachment structure

```
<Attachment [Attachment_Attributes]/>
```

### L5K ATTACHMENT structure

```
ATTACHMENT (Attachment_Attributes)
END_ATTACHMENT
```

### Attachment attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| FromID | FromElementID | The ID of the attached object. Type an unsigned, 32-bit integer value. |
| ToID | ToElementID | The ID of the object that the FromID object is attached to. Type an unsigned, 32-bit integer value. |

### Attachment guidelines

Use these guidelines for attachments:

- Use an attachment to link a text box to an FBD element.
- All attachment blocks must come after all text box blocks.

# Examples

**L5X Function Block Diagram Example**

```
- <Routine Name="MainFBD" Type="FBD">
  - <FBDContent SheetSize="Tabloid - 11 x 17 in" SheetOrientation="Landscape">
    - <Sheet Number="1">
      - <Description>
          <![CDATA[ Level_control_and_simulation ]]>
        </Description>
        <IRef ID="0" X="160" Y="420" Operand="FlowIntoTank"
          HideDesc="false" />
        <OCon ID="1" X="520" Y="280" Name="TankLevel" />
        <Block Type="ADD" ID="2" X="300" Y="100" Operand="ADD_01"
          VisiblePins="SourceA SourceB Dest" HideDesc="false" />
      - <Block Type="DEDT" ID="3" X="40" Y="100" Operand="DEDT_01"
          VisiblePins="In Out" HideDesc="false">
          <Array Name="StorageArray" Operand="DEDT_01array" />
        </Block>
        <Block Type="HLL" ID="4" X="520" Y="100" Operand="HLL_01"
          VisiblePins="In Out HighAlarm LowAlarm" HideDesc="false" />
        <Block Type="LDLG" ID="5" X="40" Y="280" Operand="LDLG_01"
          VisiblePins="In Out" HideDesc="false" />
        <Block Type="MUL" ID="6" X="480" Y="400" Operand="MUL_01"
          VisiblePins="SourceA Dest" HideDesc="false" />
        <Block Type="PIDE" ID="7" X="760" Y="60" Operand="LevelController"
          VisiblePins="PV SPProg SPCascade RatioProg CVProg FF HandFB
          ProgProgReq ProgOperReq ProgCasRatReq ProgAutoReq
          ProgManualReq ProgOverrideReq ProgHandReq CVEU SP
          PVHHAlarm PVHAlarm PVLAlarm PVLLAlarm PVROCPosAlarm
          PVROCNegAlarm DevHHAlarm DevHAlarm DevLAlarm DevLLAlarm
          ProgOper CasRat Auto Manual Override Hand" HideDesc="false" />
        <Block Type="SUB" ID="8" X="280" Y="380" Operand="SUB_01"
          VisiblePins="SourceA SourceB Dest" HideDesc="false" />
        <Wire FromID="0" ToID="8" ToParam="SourceA" />

        <Wire FromID="8" FromParam="Dest" ToID="6" ToParam="SourceA" />
        <FeedbackWire FromID="4" FromParam="Out" ToID="2"
          ToParam="SourceB" />
        <FeedbackWire FromID="7" FromParam="CVEU" ToID="3"
          ToParam="In" />
      </Sheet>
    - <Sheet Number="2">
      - <Description>
          <![CDATA[ Agitator_control ]]>
        </Description>
        <IRef ID="9" X="140" Y="300" Operand="20" HideDesc="false" />
        <ICon ID="10" X="160" Y="140" Name="TankLevel" />
        <Block Type="D2SD" ID="11" X="440" Y="220" Operand="TankAgitator"
          VisiblePins="ProgCommand State0Perm State1Perm FB0 FB1
          HandFB ProgProgReq ProgOperReq ProgOverrideReq
          ProgHandReq Out Device0State Device1State CommandStatus
          FaultAlarm ModeAlarm ProgOper Override Hand"
          HideDesc="false" />
        <Block Type="GRT" ID="12" X="240" Y="220" Operand="GRT_01"
          VisiblePins="SourceA SourceB Dest" HideDesc="false" />
        <Wire FromID="9" ToID="12" ToParam="SourceB" />
        <Wire FromID="10" ToID="12" ToParam="SourceA" />
        <Wire FromID="12" FromParam="Dest" ToID="11"
          ToParam="ProgCommand" />
        <Function Type="ADD__F" ID="3" X="500" Y="320"/>
      </Sheet>
    + <Sheet Number="3">
    </FBDContent>
  </Routine>
```

**L5K FBD_ROUTINE Example**

```
FBD_ROUTINE My_FBD_Routine (SheetSize := "Tabloid (11x17in)",
SheetOrientation := Landscape)
       SHEET  (Name := Input_Scaling)
              IREF  (ID := 3,
                     X := 120,
                     Y := 120,
                     Operand := Input_Tag)
              END_IREF

              OREF  (ID := 5,
                     X := 520,
                     Y := 320,
                     Operand := Output_Tag)
              END_OREF

              ICON  (ID := 4,
                     X := 160,
                     Y := 320,
                     Name := ConnectorName)
              END_ICON

              OCON  (ID := 6,
                     X := 680,
                     Y := 100,
                     Name := ConnectorName)
              END_OCON

              MUL_BLOCK  (ID := 0,
                          X := 440,
                          Y := 60,
                          Operand := MUL_01,
                          VisiblePins := "SourceA, SourceB,
Dest")
              END_MUL_BLOCK

              SCL_BLOCK  (ID := 1,
                          X := 240,
                          Y := 60,
                          Operand := SCL_01,
                          VisiblePins := "In, InEUMax, Out,
MaxAlarm")
              END_SCL_BLOCK

              PI_BLOCK  (ID := 2,
                         X := 260,
                         Y := 260,
                         Operand := PI_01,
                         VisiblePins := "In, Initialize,
InitialValue, Out, HighAlarm, LowAlarm")
```

```
            END_PI_BLOCK

            WIRE  (FromElementID := 3,
                   FromParameter := "",
                   ToElementID := 1,
                   ToParameter := In)
            END_WIRE

            WIRE  (FromElementID := 4,
                   FromParameter := "",
                   ToElementID := 2,
                   ToParameter := In)
            END_WIRE

            WIRE  (FromElementID := 0,
                   FromParameter := Dest,
                   ToElementID := 6,
                   ToParameter := "")
            END_WIRE

            WIRE  (FromElementID := 1,
                   FromParameter := Out,
                   ToElementID := 0,
                   ToParameter := SourceA)
            END_WIRE

            WIRE  (FromElementID := 2,
                   FromParameter := Out,
                   ToElementID := 5,
                   ToParameter := "")
            END_WIRE

            FEEDBACK_WIRE  (FromElementID := 0,
                            FromParameter := Dest,
                            ToElementID := 0,
                            ToParameter := SourceB)
            END_FEEDBACK_WIRE

            ADD_FUNCTION  (ID := 13,
                           X := 340,
                           Y := 120)
            END_ADD_FUNCTION

        END_SHEET

    END_FBD_ROUTINE
```

# Parameters for function block instructions

These tables list each function block instruction and its format in the Block component of an import/export file. For details about a specific instruction, see the listed reference manuals.

**Function block instruction reference manuals**

| Instruction Type | Resource |
|---|---|
| Basic, sequential instruction | Logix5000 Controllers General Instructions Set Reference Manual, publication 1756-RM003. |
| Process control or drives instruction | Logix5000 Controllers Advanced Process Control and Drives Instruction Set Reference Manual, publication1756-RM006. |
| Motion instruction | Logix5000 Controllers Motion Instructions Reference Manual, publication -RM002. |

**Function block instructions**

| Instruction | Default Operand and VisiblePins formats (components within the Block structure) |
|---|---|
| ABS | Operand := ABS_01,<br>VisiblePins := "*Source, Destination*") |
| ACS | Operand := ACS_01,<br>VisiblePins := "*Source, Destination*") |
| ADD | Operand := ADD_01,<br>VisiblePins := "*SourceA, SourceB, Destination*") |
| ALM | Operand := ALM_01,<br>VisiblePins := "*In, HHAlarm, HAlarm, LAlarm, LLAlarm, ROCPosAlarm, ROCNegAlarm*") |
| ALMA | Operand := ALMA_01,<br>VisiblePins := "*In, HHInAlarm, HInAlarm, LInAlarm, LLInAlarm, ROCPosInAlarm, ROCNegInAlarm, HHAcked, HAcked, LAcked, LLAcked, ROCPosAcked, ROCNegAcked, Suppressed, Disabled*") |
| ALMD | Operand := ALMD_01,<br>VisiblePins := "*In, InAlarm, Acked, Suppressed, Disabled*") |
| AND | Operand := AND_01,<br>VisiblePins := "*SourceA, SourceB, Destination*") |
| ASN | Operand := ASN_01,<br>VisiblePins := "*Source, Destination*") |
| ATN | Operand := ATN_01,<br>VisiblePins := "*Source, Destination*") |
| BAND | Operand := BAND_01,<br>VisiblePins := "*In1, In2, In3, In4, Out*") |
| BNOT | Operand := BNOT_01,<br>VisiblePins := "*In, Out*") |
| BOR | Operand := BOR_01,<br>VisiblePins := "*In1, In2, In3, In4, Out*") |
| BTDT | Operand := BTDT_01,<br>VisiblePins := *"Source, SourceBit, Length, DestBit, Target, Dest"*) |
| BXOR | Operand := BXOR_01,<br>VisiblePins := "*In1, In2, Out*") |

| Instruction | Default Operand and VisiblePins formats (components within the Block structure) |
|---|---|
| COS | Operand := COS_01,<br>VisiblePins := "*Source, Dest*") |
| CTUD | Operand := CTUD_01,<br>VisiblePins := "*CUEnable, CDEnable, PRE, Reset, ACC, DN*") |
| D2SD | Operand := D2SD_01,<br>VisiblePins := "*ProgCommand, State0Perm, State1Perm, FB0, FB1, HandFB, ProgProgReq, ProgOperReq, ProgOverrideReq, ProgHandReq, Out, Device0State, Device1State, CommandStatus, FaultAlarm, ModeAlarm, ProgOper, Override, Hand*") |
| D3SD | Operand := D3SD_01,<br>VisiblePins := "*Prog0Command, Prog1Command, Prog2Command, State0Perm, State1Perm, State2Perm, FB0, FB1, FB2, FB3, HandFB0, HandFB1, HandFB2, ProgProgReq, ProgOperReq, ProgOverrideReq, ProgHandReq, Out0, Out1, Out2, Device0State, Device1State, Device2State, Command0Status, Command1Status, Command2Status, FaultAlarm, ModeAlarm, ProgOper, Override, Hand*") |
| DEDT | Operand := DEDT_01,<br>VisiblePins := "*In, Out*",<br>Storage := *array_name*) |
| DEG | Operand := DEG_01,<br>VisiblePins := "*Source, Dest*") |
| DERV | Operand := DERV_01,<br>VisiblePins := "*In, ByPass, Out*") |
| DFF | Operand := DFF_01,<br>VisiblePins := "*D, Clear, Clock, Q, QNot*") |
| DIV | Operand := DIV_01,<br>VisiblePins := "*SourceA, SourceB, Dest*") |
| ESEL | Operand := ESEL_01,<br>VisiblePins := "*In1, In2, In3, In4, In5, In6, ProgSelector, ProgProgReq, ProgOperReq, ProgOverrideReq, Out, SelectedIn, ProgOper, Override*") |
| EQU | Operand := EQU_01,<br>VisiblePins := "*SourceA, SourceB*") |
| FGEN | Operand := FGEN_01,<br>VisiblePins := "*In, Out*",<br>X1 := *array_name*,<br>X2 := *array_name*,<br>Y2 := *array_name*,<br>Y2 := *array_name*) |
| FRD | Operand := FRD_01,<br>VisiblePins := "*Source, Dest*") |
| GEQ | Operand := GEQ_01,<br>VisiblePins := "*SourceA, SourceB*") |
| GRT | Operand := GRT_01,<br>VisiblePins := "*SourceA, SourceB*") |
| HLL | Operand := HLL_01,<br>VisiblePins := "*In, Out, HighAlarm, LowAlarm*") |
| HPF | Operand := HPF_01,<br>VisiblePins := "*In, Out*") |
| INTG | Operand := INTG_01,<br>VisiblePins := "*In, Out*") |
| JKFF | Operand := JKFF_01,<br>VisiblePins := "*Clear, Clock, Q, QNot*") |

| Instruction | Default Operand and VisiblePins formats (components within the Block structure) |
|---|---|
| LEQ | Operand := LEQ_01,<br>VisiblePins := "*SourceA, SourceB*") |
| LES | Operand := LES_01,<br>VisiblePins := "*SourceA, SourceB*") |
| LIM | Operand := LIM_01,<br>VisiblePins := "*LowLlimit, Test, HighLimit*") |
| LN | Operand := LN_01,<br>VisiblePins := "*Source, Dest*") |
| LOG | Operand := LOG_01,<br>VisiblePins := "*Source, Dest*") |
| LPF | Operand := LPF_01,<br>VisiblePins := "*In, Out*") |
| MAVE | Operand := MAVE_01,<br>VisiblePins := "*In, Out*",<br>Storage := *array_name*,<br>Weight := *array_name*) |
| MAXC | Operand := MAXC_01,<br>VisiblePins := "*In, Reset, ResetValue, Out*") |
| MEQ | Operand := MEQ_01,<br>VisiblePins := "*Source, Mask, Compare*") |
| MINC | Operand := MINC_01,<br>VisiblePins := "*In, Reset, ResetValue, Out*") |
| MOD | Operand := MOD_01,<br>VisiblePins := "*SourceA, SourceB, Dest*") |
| MSTD | Operand := MSTD_01,<br>VisiblePins := "*In, SampleEnable, Out*",<br>Storage := *array_name*) |
| MUL | Operand := MUL_01,<br>VisiblePins := "*SourceA, SourceB, Dest*") |
| MUX | Operand := MUX_01,<br>VisiblePins := "*In1, In2, In3, In4, In5, In6, In7, In8, Selector, Out*") |
| MVMT | Operand := MVMT_01,<br>VisiblePins := "*Source, Mask, Target, Dest*") |
| NEG | Operand := NEG_01,<br>VisiblePins := "*Source, Dest*") |
| NEQ | Operand := NEQ_01,<br>VisiblePins := "*SourceA, SourceB*") |
| NOT | Operand := NOT_01,<br>VisiblePins := "*Source, Dest*") |
| NTCH | Operand := NTCH_01,<br>VisiblePins := "*In, Out*") |
| OR | Operand := OR_01,<br>VisiblePins := "*SourceA, SourceB, Dest*") |
| OSFI | Operand := OSFI_01,<br>VisiblePins := "*InputBit, OutputBit*") |

| Instruction | Default Operand and VisiblePins formats (components within the Block structure) |
|---|---|
| OSRI | Operand := OSRI_01,<br>VisiblePins := "*InputBit, OutputBit*") |
| PI | Operand := PI_01,<br>VisiblePins := "*In, Out*") |
| PIDE | Operand := PIDE_01,<br>VisiblePins := "*PV, SPProg, SPCascade, RatioProg, CVProg, FF, HandFB, ProgProgReq, ProgOperReq, ProgCasRatReq, ProgAutoReq, ProgManuaReq, ProgOverrideReq, ProgHandReq, CVEU, SP, PVHHAlarm, PVHAlarm, PVLAlarm, PVLLAlarm, PVROCPosAlarm, PVROCNegAlarm, DevHHAlarm, DevHAlarm, DevLAlarm, DevLLAlarm, ProgOper, CasRat, Auto, Manual, Override, Hand*") |
| PMUL | Operand := PMUL_01,<br>VisiblePins := "*In, Multipler, Out*") |
| POSP | Operand := POSP_01,<br>VisiblePins := "*SP, Position, OpenedFB, ClosedFB, OpenOut, CloseOut*") |
| RAD | Operand := RAD_01,<br>VisiblePins := "*Source, Dest*") |
| RESD | Operand := RESD_01,<br>VisiblePins := "*Set, Reset, Out, OutNot*") |
| RLIM | Operand := RLIM_01,<br>VisiblePins := "*In, ByPass, Out*") |
| RMPS | Operand := RMPS_01,<br>VisiblePins := "*PV, CurrentSegProg, OutProg, SoakTimeProg, ProgProgReq, ProgOperReq, ProgAutoReq, ProgManualReq, ProgHoldReq, Out, CurrentSeg, SoakTimeLeft, GuarRampOn, GuarSoakOn, ProgOper, Auto, Manual, Hold*",<br>RampValue := *array_name*,<br>SoakValue := *array_name*,<br>SoakTime := *array_name*) |
| RTOR | Operand := RTOR_01,<br>VisiblePins := *TimerEnable, PRE, Reset, ACC, DN")* |
| SCL | Operand := SCL_01,<br>VisiblePins := "*In, Out*") |
| SCRV | Operand := SCRV_01,<br>VisiblePins := "*In, Out*") |
| SEL | Operand := SEL_01,<br>VisiblePins := "*In1, In2, SelectorIn, Out*") |
| SETD | Operand := SETD_01,<br>VisiblePins := "*Set, Reset, Out, OutNot*") |
| SIN | Operand := SIN_01,<br>VisiblePins := SIN(*source,destination*); |
| SNEG | Operand := SNEG_01,<br>VisiblePins := "*In, NegateEnable, Out*") |
| SOC | Operand := SOC_01,<br>VisiblePins := "*In, Out*") |
| SQR | Operand := SQR_01,<br>VisiblePins := "*Source, Dest*") |
| SRTP | Operand := SRTP_01,<br>VisiblePins := "*In, HeatOut, CoolOut, HeatTimePercent, CoolTimePercent*") |
| SSUM | Operand := SSUM_01,<br>VisiblePins := "*In1, Select1, In2, Select2, In3, Select3, In4, Select4, Out*") |

| Instruction | Default Operand and VisiblePins formats (components within the Block structure) |
|---|---|
| SUB | Operand := SUB_01,<br>VisiblePins := "*SourceA, SourceB, Dest*") |
| TAN | Operand := TAN_01,<br>VisiblePins := "*Source, Dest")* |
| TOD | Operand := TOD_01,<br>VisiblePins := "*Source, Dest")* |
| TOFR | Operand := TOFR_01,<br>VisiblePins := "*TimerEnable, PRE, Reset, ACC, DN*") |
| TONR | Operand := TONR_01,<br>VisiblePins := "*TimerEnable, PRE, Reset, ACC, DN*") |
| TOT | Operand := TOT_01,<br>VisiblePins := "*In, ProgProgReq, ProgOperReq, ProgStartReq, ProgStopReq, ProgResetReq, Total, OldTotal, ProgOper, RunStop, ProgResetDone, TargetFlag, TargetDev1Flag, TargetDev2Flag*") |
| TRN | Operand := TRN_01,<br>VisiblePins := "*Source, Dest")* |
| UPDN | Operand := UPDN_01,<br>VisiblePins := "*InPlus, InMinus, Out*") |
| XOR | Operand := XOR_01,<br>VisiblePins := "*SourceA, SourceB, Dest*") |
| XPY | Operand := XPY_01,<br>VisiblePins := "*SourceA, SourceB, Dest*") |

# Define a sequential function chart routine

## Introduction

This chapter explains how to enter sequential function chart logic in a complete import/export file.

For more information on creating sequential function charts and correct syntax, see the Logix5000 Controller Common Procedures Programming Manual, publication 1756-PM001.

## Sequential function chart routine

These examples show sequential function chart structure.

### L5X sequential function chart structure

```
<Routines>
      <Routine [Routine_Attributes]>
            <Description>
                  <![CDATA[ text ]]>
            </Description>
            <SFCContent [SFCContent_Attributes]>
                  <Step [Step_Attributes]/>
                  <Transition [Transition_Attributes]/>
                  <Branch [Branch_Attributes]/>
                  <SbrRet [SbrRet_Attributes]/>
                  <Stop [Stop_Attributes]/>
                  <Branch [Branch_Attributes]/>
                  <DirectedLink [DirectedLink_Attributes]/>
                  <TextBox [TextBox_Attributes]>
                        text
                  </TextBox>
                  <Attachment [Attachment_Attributes]/>
            </SFCContent>
      </Routine>
</Routines>
```

### L5K sequential function chart SFC_ROUTINE structure

```
SFC_ROUTINE <routine_name> [(Routine_Attributes,
            SFC_Attributes)]
      <STEP declaration>
      <TRANSITION declaration>
      <BRANCH declaration>
      <SBR_RET declaration>
      <STOP declaration>
      <BRANCH declaration>
```

```
        <DIRECTED_LINK declaration>
        <TEXT_BOX declaration)
        <ATTACHMENT declaration>
END_SFC_ROUTINE
```

## Sequential function chart elements

| L5X Item | L5K Item | Identifies |
|---|---|---|
| N/A | routine_name | The name of the routine.<br><br>In L5X, use a Name attribute on the <Routine> element. |
| Description | Description | User information about the routine. |
| SFCContent | N/A | Sequential function chart logic. |
| EncryptionInfo | ENCRYPTION_INFO | Details of the license-based Source Protection for the lockable object. Only exists for protected routines exported in plain text. |
| EncryptedContent | N/A | Source Protected and locked routine content. Only exists for locked routines exported in plain text. |
| EncryptedSegments | N/A | Locked logic for the routine. Only exists for locked routines exported in plain text. |

## SFC_Routine attributes

| Attribute | Description |
|---|---|
| Name | L5X only. Specify the name of the routine.<br><br>For L5K, the name is an element of the statement. |
| Type | L5X only. Specify SFC.<br><br>In L5K, the type of routine is part of the routine statement. |
| SheetSize | The size of the SFC. Select one of these options.<br>• Letter (8.5x11in)<br>• Legal (8.5x14in)<br>• Tabloid (11x17in)<br>• A4 (210x297mm)<br>• A3 (297x420mm)<br><br>In L5X, this attribute is on the <SFCContent> element. |
| SheetOrientation | The orientation of the SFC sheet. Select **Portrait** or **Landscape**.<br><br>In L5X, this attribute is on the <SFCContent> element. |

| Attribute | Description |
|---|---|
| StepName | The prefix for the name of the step blocks within this SFC routine. The Logix Designer application uses this prefix when it automatically generates an SFC_STEP tag.<br><br>In L5X, this attribute is on the <SFCContent> element. |
| TransitionName | The prefix for the name of the transition blocks with this SFC routine. The Logix Designer application uses this prefix when it automatically generates a transition tag.<br><br>In L5X, this attribute is on the <SFCContent> element. |
| ActionName | The prefix for the name of the action blocks in this SFC routine. The Logix Designer application uses this prefix when it automatically generates an SFC_ACTION tag.<br><br>In L5X, this attribute is on the <SFCContent> element. |
| StopName | The prefix for the name of the stop blocks in this SFC routine. The Logix Designer application uses this prefix when it automatically generates an SFC_STOP tag.<br><br>In L5X, this attribute is on the <SFCContent> element. |
| OnlineEditType | L5X only. Specify the online edit logic type (**Original**, **PendingEdits**, or **TestEdits**). This attribute is not specified if there are no edits.<br><br>In L5X, this attribute is on the <SFCContent> element. |
| PermissionSet | Name of the set of permissions, configured in FactoryTalk Security, to apply to this object. |
| TrackingGroups | The group of tracked objects to which this item belongs. Components can be marked for tracking to determine whether they have been changed. Version 30 of the Logix Designer application supports only one tracking group. |

## Export sequential function chart logic while editing online

If you export sequential function chart logic that contains online edits, the export file exports LOGIC blocks (in L5K format) or additional <SFCContent> elements (in L5X format) to indicate the original, test edits, and pending edits states. If there are no online edits, you will not see these LOGIC blocks or additional <SFCContent> elements.

### L5X Example: Test edits and pending edits exist

```
- <Routine Name="SFCRoutine" Type="SFC">
  - <SFCContent SheetSize="Letter - 8.5 x 11 in" SheetOrientation="Landscape" StepName="Step"
    TransitionName="Tran" ActionName="Action" StopName="Stop" OnlineEditType="Original">
      <!--  SFC content inserted here - see format descibed above  -->
    </SFCContent>
  - <SFCContent SheetSize="Letter - 8.5 x 11 in" SheetOrientation="Landscape" StepName="Step"
    TransitionName="Tran" ActionName="Action" StopName="Stop" OnlineEditType="PendingEdits">
      <!--  SFC content inserted here - see format descibed above  -->
    </SFCContent>
  - <SFCContent SheetSize="Letter - 8.5 x 11 in" SheetOrientation="Landscape" StepName="Step"
    TransitionName="Tran" ActionName="Action" StopName="Stop" OnlineEditType="TestEdits">
      <!--  SFC content inserted here - see format descibed above  -->
    </SFCContent>
  </Routine>
```

### L5K Example 1: Both test edits and pending edits exist

```
SFC_ROUTINE MySFCRoutine (SheetSize := "Letter (8.5x11in)",
       SheetOrientation := Landscape, StepName := "Step",
       TransitionName := "Tran", ActionName := "Action",
       StopName := "Stop")
       LOGIC (Online_Edit_Type := Orig)
              (* SFC logic here *)
       END_LOGIC

       LOGIC (Online_Edit_Type := Test)
              (* SFC logic here *)
       END_LOGIC

       LOGIC (Online_Edit_Type := Pend)
              (* SFC logic here *)
       END_LOGIC
END_SFC_ROUTINE
```

### L5K Example 2: Only pending edits exist

```
SFC_ROUTINE MySFCRoutine (SheetSize := "Letter (8.5x11in)",
       SheetOrientation := Landscape, StepName := "Step",
       TransitionName := "Tran", ActionName := "Action",
       StopName := "Stop")
       LOGIC (Online_Edit_Type := Orig)
              (* SFC logic here *)
       END_LOGIC

       LOGIC (Online_Edit_Type := Pend)
              (* SFC logic here *)
       END_LOGIC
END_SFC_ROUTINE
```

**Online edit types**

| Item | Identifies |
|---|---|
| Online_Edit_Type | When you export logic:<br>• If online edits exist, there is a LOGIC block for Online_Edit_Type := Orig and the appropriate LOGIC block for the existing edits. Online_Edit_Type : = Pend indicates pending edits. Online_Edit_Type := Test indicates test edits.<br>• If there are no online edits when you export the logic, there are no LOGIC blocks and the main components in the routine are SFC logic components. |

# Steps

These examples show the step structure.

## L5X step structure

```
<Step [Step_Attributes]>
      <Preset>
            logic
      </Preset>
      <LimitHigh>
            logic
      </LimitHigh>
      <LimitLow>
            logic
      </LimitLow>
      <Action>
            logic
      </Action>
</Step>
```

## L5K STEP structure

```
STEP (Step_Attributes)
      <PRESET declaration>
      <LIMIT_HIGH declaration>
      <LIMIT_LOW declaration>
      <ACTION_LIST declaration>
END_STEP
```

## Step elements

| L5X Item | L5K Item | Identifies |
|---|---|---|
| Preset | PRESET | A structured text expression that specifies the preset time in milliseconds for the step timer. If the PresetUsesExpression attribute (above) is **Yes**, type preset logic. |
| LimitHigh | LIMIT_HIGH | A structured text expression that specifies the preset time in milliseconds for a limit high alarm. If the LimitHighUsesExpression attribute (above) is **Yes**, type limit high logic. |
| LimitLow | LIMIT_LOW | A structured text expression that specifies the preset time in milliseconds for a limit low alarm. If the LimitLowUsesExpression attribute (above) is **Yes**, type limit low logic. |
| ActionList | ACTION_LIST | The actions in the step. |

## Step attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| ID | ID | The step identifier. This ID uniquely identifies this step from all other blocks. Type an unsigned, 32-bit integer value. |
| X | X | X-coordinate on internal grid. Type an unsigned, 32-bit integer value. |
| Y | Y | Y-coordinate on internal grid. Type an unsigned, 32-bit integer value. |
| Operand | Operand | The step tag. Type a tag of datatype SFC_STEP. The import process uses this tag name to name the step. |
| HideDesc | HideDescription | Whether or not to hide the step description. Type **Yes** or **No**. |
| DescX | DescriptionX | X-coordinate on internal grid of the description box. Type an unsigned, 32-bit integer value. |
| DescY | DescriptionY | Y-coordinate on internal grid of the description box. Type unsigned, 32-bit integer value. |
| DescWidth | DescriptionWidth | This attribute is not currently used; it is there for future use. Type **0**. |
| InitialStep | InitialStep | Whether this step is the initial step of the routine. Type **Yes** or **No**.<br>If you have multiple steps identified as the initial step, which is incorrect syntax, the import process designates the last initial step it encounters as the initial step and removes the initial step indicators from any other steps. |
| PresetUsesExpr | PresetUsesExpression | Whether the preset for the step timer is a structured text expression. Type **Yes** if you plan to enter an expression in a preset element, otherwise, type **No**. |
| LimitHighUsesExpr | LimitHighUsesExpression | Whether the preset for the limit high alarm is a structured text expression. Type **Yes** if you plan to enter an expression in a limit high element, otherwise, type **No**. |
| LimitLowUsesExpr | LimitLowUsesExpression | Whether the preset for the limit low alarm is a structured text expression. Type **Yes** if you plan to enter an expression in a limit low element, otherwise, type **No**. |
| ShowActions | ShowActions | Whether to show or hide the step's actions. Type **Yes** or **No**. |

# Preset

The preset component contains a structured text expression that specifies the preset time in milliseconds for the step timer.

## L5X preset structure

```
<Preset>
      <STContent>
            <Line Number="number">
                  <![CDATA[ structured_text; ]]>
            </Line>
      </STContent>
</Preset>
```

## L5K PRESET structure

```
PRESET (LanguageType := ST)
      '<structured_text>
END_PRESET
```

Each line of L5K structured text begins with a single quote (').

## Limit high

The limit high component contains a structured text expression that specifies the preset time in milliseconds for a limit high alarm.

### L5X limit high structure

```
<LimitHigh>
      <STContent>
            <Line Number="0">
                  <![CDATA[ structured_text; ]]>
            </Line>
      </STContent>
</LimitHigh>
```

### L5K LIMITHIGH structure

```
LIMITHIGH (LanguageType := ST)
      ‘<structured_text>
END_LIMITHIGH
```

Each line of L5K structured text begins with a single quote (‘).

## Limit low

The limit low component contains a structured text expression that specifies the preset time in milliseconds for a limit low alarm.

### L5X limit low structure

```
<LimitLow>
      <STContent>
            <Line Number="0">
                  <![CDATA[ structured_text; ]]>
            </Line>
      </STContent>
</LimitLow>
```

### L5K LIMITHIGH structure

```
LIMITLOW (LanguageType := ST)
      ‘<structured_text>
END_LIMITLOW
```

Each line of L5K structured text begins with a single quote (‘).

## Action list

Each step can contain multiple actions.

### L5X Action Structure

```
<Action [Action_Attributes]>
      <Preset>
            logic
      </Preset>
      <Body>
            logic
      </Body>
</Action>
```

### L5K ACTION structure

```
ACTION (Action_Attributes)
      <PRESET declaration>
      <BODY declaration>
END_ACTION
```

### Action attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| ID | ID | The action identifier. This ID uniquely identifies this action from all other blocks. Enter an unsigned, 32-bit integer value. |
| Operand | Operand | The action tag. Enter a tag of datatype SFC_ACTION. The import process uses this tag name to name the action. |
| Qualifier | Qualifier | The action qualifier.<br>**Qualifier** — **Description**<br>N — non-stored<br>R — reset<br>S — stored<br>L — time limited<br>D — time delayed<br>P — pulse<br>P1 — pulse (rising edge)<br>P0 — pulse (falling edge)<br>SL — stored and time limited<br>SD — stored and time delayed<br>DS — time delayed and stored |
| IsBoolean | IsBoolean | Whether or not the action is boolean. Enter Yes or No. |
| PresetUsesExpr | PresetUsesExpression | Whether the preset for the action timer is a structured text expression. Enter Yes if you plan to enter an expression in a PRESET block, otherwise, enter No. |
| IndicatorTag | IndicatorTag | The indicator tag. Enter tag. |

## Transitions

These examples show transition structure.

### L5X transition structure

```
<Transition [Transition_Attributes]>
      <Condition>
            logic
      </Condition>
</Transition>
```

### L5K TRANSITION structure

```
TRANSITION (Transition_Attributes)
      <CONDITION declaration>
END_TRANSITION
```

### Transition elements

| L5X Item | L5K Item | Identifies |
|---|---|---|
| Condition | CONDITION | The condition to evaluate for the transition. |

## Transition attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| ID | ID | The step identifier. This ID uniquely identifies this step from all other blocks. Type an unsigned, 32-bit integer value. |
| X | X | X-coordinate on internal grid. Type an unsigned, 32-bit integer value. |
| Y | Y | Y-coordinate on internal grid. Type an unsigned, 32-bit integer value. |
| Operand | Operand | The step tag. Type a tag of datatype SFC_STEP. The import process uses this tag name to name the step. |
| HideDesc | HideDescription | Whether or not to hide the step description. Type **Yes** or **No**. |
| DescX | DescriptionX | X-coordinate on internal grid of the description box. Type an unsigned, 32-bit integer value. |
| DescY | DescriptionY | Y-coordinate on internal grid of the description box. Type unsigned, 32-bit integer value. |
| DescWidth | DescriptionWidth | This attribute is not currently used; it is there for future use. Type **0**. |
| Force | Force | The transition is forced. Type **true** for forced true (set) or type **false** for forced false (cleared). If the transition is not forced, do not enter this attribute. |

# Condition

The condition component uses a structured text expression to specify a condition to evaluate for the transition.

## L5X condition structure

```
<Condition>
      <STContent>
            <Line Number="0">
                  <![CDATA[ structured_text; ]]>
            </Line>
      </STContent>
</Condition>
```

## L5K CONDITION structure

```
CONDITION (LanguageType := ST)
      ‘<structured_text>
END_CONDITION
```

Each line of L5K structured text begins with a single quote (‘).

# Subroutine calls

Subroutine calls pass values into and out of the SFC routine.

## L5X SbrRet structure

```
<SbrRet [Subroutine_Attributes]/>
```

## L5K SBR_RET structure

```
SBR_RET (Subroutine_Attributes)
END_SBR_RET
```

## Subroutine attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| ID | ID | The SBR_RET identifier. This ID uniquely identifies this subroutine call from all other blocks. Type an unsigned, 32-bit integer value. |
| X | X | X-coordinate on internal grid. Type an unsigned, 32-bit integer value. |
| Y | Y | Y-coordinate on internal grid. Type an unsigned, 32-bit integer value. |
| In | In | List of values to receive from the calling routine. Type list of tags or literal values and separate each entry by a comma (,). Enter empty quotes if there are no values to receive. |
| Ret | Out | List of values to pass to the calling routine. Type list of tags or literal values and separate each entry by a comma (,). Enter empty quotes if there are no values to pass. |

# Stops

These examples show stop structure.

## L5X stop structure

```
<Stop [StopAttributes]/>
```

## L5K STOP structure

```
STOP (Stop_Attributes)
END_STOP
```

## Stop attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| ID | ID | The stop identifier. This ID uniquely identifies this stop from all other blocks. Type an unsigned, 32-bit integer value. |
| X | X | X-coordinate on internal grid. Type an unsigned, 32-bit integer value. |
| Y | Y | Y-coordinate on internal grid. Type an unsigned, 32-bit integer value. |
| Operand | Operand | The stop tag. Type a tag of datatype SFC_STOP. The import process uses this tag name to name the stop. |
| HideDesc | HideDescription | Whether or not to hide the stop description. Type **Yes** or **No**. |
| DescX | DescriptionX | X-coordinate on internal grid of the description box. Type an unsigned, 32-bit integer value. |
| DescY | DescriptionY | Y-coordinate on internal grid of the description box. Type unsigned, 32-bit integer value. |
| DescWidth | DescriptionWidth | This attribute is not currently used; it is there for future use. Type **0**. |

# Branches

The branch blocks in an SFC routine identify simultaneous or selection branches in the routine.

## L5X branch structure

```
<Branch [BranchAttributes]>
      <Leg [LegAttributes]/>
</Branch>
```

## L5K BRANCH structure

```
BRANCH (Branch_Attributes)
      LEG (Leg_Attributes)
      END_LEG
END_BRANCH
```

### Branch attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| ID | ID | The branch identifier. This ID uniquely identifies this branch from all other blocks. Type an unsigned, 32-bit integer value. |
| X | X | X-coordinate on internal grid. Type an unsigned, 32-bit integer value. |
| Y | Y | Y-coordinate on internal grid. Type an unsigned, 32-bit integer value. |
| BranchType | BranchType | The type of branch. Type **Simultaneous** or **Selection**. |
| BranchFlow | BranchFlow | The direction of the branch. Type **Converge** or **Diverge**. |
| Priority | Priority | Whether the priority of a divergent selection branch is defined by the user. This attribute applies only to divergent selection branches. Type **Default** or **UserDefined**. |

### Leg attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| ID | ID | The leg identifier. This ID uniquely identifies this leg from all other blocks. Type an unsigned, 32-bit integer value. |
| Force | Force | Whether the leg is forced or not. You can force only a leg in a simultaneous branch. Omit this attribute, for no forces, or type **false** to force the leg false. |

## Directed links

The directed link blocks in an SFC routine identify the links between SFC components.

### L5X DirectedLink structure

```
<DirectedLink  [DirectedLinkAttributes]/>
```

### L5K DIRECTED_LINK structure

```
DIRECTED_LINK (Directed_Link_Attributes)
END_DIRECTED_LINK
```

### Directed link attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| FromID | FromElementID | The ID of the object. Type an unsigned, 32-bit integer value. |
| ToID | ToElementID | The ID of the object that the FromID object is attached to. Type an unsigned, 32-bit integer value. |
| Show | ShowLink | Whether or not to show the link. Type **TRUE** or **FALSE**. |

### Directed link guidelines

Use these guidelines for directed links:

- All directed link blocks must come after all step, transition, stop, and branch blocks.

- A directed link links only one element to one other element.

## Text boxes

The text box blocks in an SFC routine hold descriptions about SFC components.

### L5X TextBox structure

```
<TextBox [TextBoxAttributes]>
      <![CDATA[ text  ]] >
</TextBox>
```

### L5K TEXTBOX structure

```
TEXT_BOX (Text_Box_Attributes,
      Text := <"text">)
END_TEXT_BOX
```

### Text box attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| ID | ID | The element identifier; uniqueness is important for wiring.<br>Type an unsigned, 32-bit integer value. |
| X | X | X-coordinates on internal grid.<br>Type an unsigned, 32-bit integer value. |
| Y | Y | Y-coordinates on internal grid.<br>Type an unsigned, 32-bit integer value. |
| Width | Width | This attribute is not currently used; it is there for future use. Type **0**. |
| Text | Text | The descriptive text. |

### Text box guidelines

Use these guidelines for text boxes:

- All text box blocks must come after all directed link blocks.
- Text boxes can be free-standing or they can be attached to SFC elements.

## Attachments

The attachment blocks in an SFC routine identify the attachments from text boxes to other SFC elements.

### L5X attachment structure

```
<Attachment [Attachment_Attributes]/>
```

### L5K ATTACHMENT structure

```
ATTACHMENT (Attachment_Attributes)
END_ATTACHMENT
```

### Attachment attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| FromID | FromElementID | The ID of the attached object. Type an unsigned, 32-bit integer value. |
| ToID | ToElementID | The ID of the object that the FromID object is attached to. Type an unsigned, 32-bit integer value. |

## Attachment guidelines

Use these guidelines for attachments:

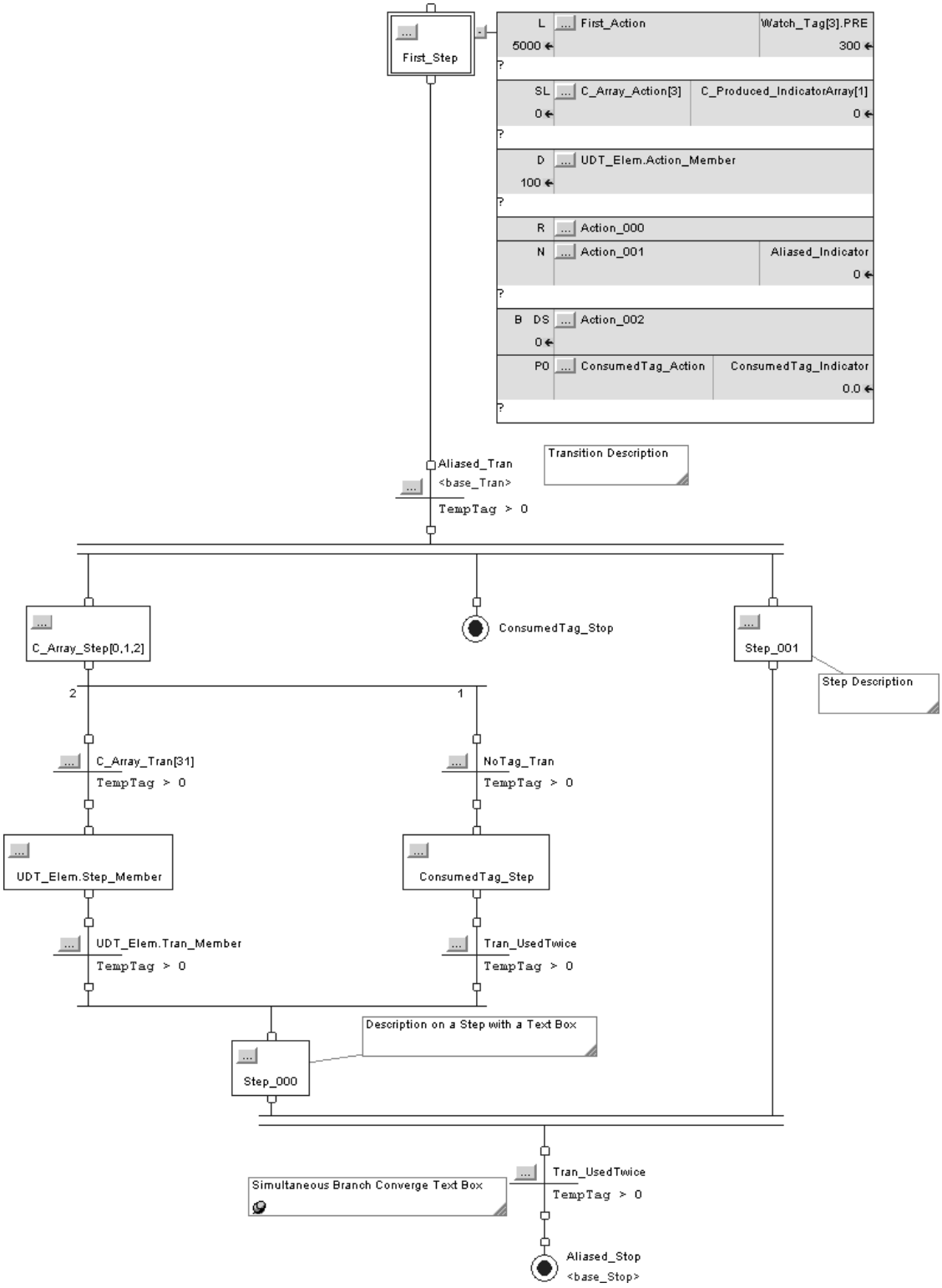
- Use an attachment to link a text box to an SFC element.
- All attachment blocks must come after all text box blocks.

## Examples

**L5X sequential function chart example**

```
- <Routine Name="SimpleMotion" Type="SFC">
  - <SFCContent SheetSize="Letter - 8.5 x 11 in" SheetOrientation="Landscape"
      StepName="Step" TransitionName="Tran" ActionName="Action"
      StopName="Stop">
    - <Step ID="0" X="240" Y="360" Operand="ServosOn" HideDesc="false"
        DescX="298" DescY="345" DescWidth="0" InitialStep="true"
        PresetUsesExpr="false" LimitHighUsesExpr="false" LimitLowUsesExpr="false"
        ShowActions="true">
      - <Action ID="1" Operand="Action_000" Qualifier="NonStored"
          IsBoolean="false" PresetUsesExpr="false">
        - <Body>
          - <STContent>
            - <Line Number="0">
                <![CDATA[ MSO(axis1, axis1_MSO);  ]]>
              </Line>
            </STContent>
          </Body>
        </Action>
      + <Action ID="2" Operand="Action_008" Qualifier="NonStored"
          IsBoolean="false" PresetUsesExpr="false">
      + <Action ID="3" Operand="Action_009" Qualifier="Pulse" IsBoolean="false"
          PresetUsesExpr="false">
      </Step>
    + <Transition ID="4" X="240" Y="480" Operand="Tran_000" HideDesc="false"
        DescX="275" DescY="465" DescWidth="0">
      <DirectedLink FromID="0" ToID="4" Show="true" />
    - <TextBox ID="5" X="560" Y="340" Width="0">
      - <Text>
          <![CDATA[ This program demonstrates how Sequential Function
          Chart programs work.  ]]>
        </Text>
      </TextBox>
    </SFCContent>
  </Routine>
```

**L5K SFC_ROUTINE Example**




```
SFC_ROUTINE Sample_SFC_Routine1 (SheetSize := "Letter
(8.5x11in)",
```

```
            SheetOrientation := Landscape, StepName :=
"Step",
            TransitionName := "Tran", ActionName := "Action",
            StopName := "Stop")
      TRANSITION  (ID := 0, X := 120, Y := 1000, Operand :=
C_Array_Tran[31],
            HideDescription := Yes, DescriptionX := 155,
DescriptionY := 985,
            DescriptionWidth := 0)
            CONDITION  (LanguageType := ST)
            'TempTag > 0
            END_CONDITION
      END_TRANSITION
      BRANCH  (ID := 2, Y := 820, BranchType := Simultaneous,
BranchFlow := Diverge)
            LEG  (ID := 3)
            END_LEG
            LEG  (ID := 4)
            END_LEG
            LEG  (ID := 5)
            END_LEG
      END_BRANCH
      TRANSITION  (ID := 6, X := 420, Y := 760, Operand :=
Aliased_Tran,
            HideDescription := No, DescriptionX := 520,
DescriptionY := 740,
            DescriptionWidth := 0)
            CONDITION  (LanguageType := ST)
            'TempTag > 0
            END_CONDITION
      END_TRANSITION
      STOP  (ID := 8, X := 460, Y := 880, Operand :=
ConsumedTag_Stop,
            HideDescription := Yes, DescriptionX := 565,
DescriptionY := 865,
            DescriptionWidth := 0)
      END_STOP
      TRANSITION  (ID := 10, X := 520, Y := 1360, Operand :=
Tran_UsedTwice,
            HideDescription := Yes, DescriptionX := 555,
DescriptionY := 1345,
            DescriptionWidth := 0)
            CONDITION  (LanguageType := ST)
            'TempTag > 0
            END_CONDITION
      END_TRANSITION
      TRANSITION  (ID := 12, X := 460, Y := 1160, Operand :=
Tran_UsedTwice,
            HideDescription := Yes, DescriptionX := 495,
DescriptionY := 1145,
            DescriptionWidth := 0)
            CONDITION  (LanguageType := ST)
```

```
            'TempTag > 0
            END_CONDITION
      END_TRANSITION

      BRANCH  (ID := 14, Y := 940, BranchType := Selection,
BranchFlow := Diverge,
            Priority := UserDefined)
            LEG  (ID := 15)
            END_LEG
            LEG  (ID := 16)
            END_LEG
      END_BRANCH
      BRANCH  (ID := 17, Y := 1320, BranchType :=
Simultaneous, BranchFlow := Converge)
            LEG  (ID := 18)
            END_LEG
            LEG  (ID := 19)
            END_LEG
      END_BRANCH
      STOP  (ID := 20, X := 520, Y := 1440, Operand :=
Aliased_Stop, HideDescription := No,
            DescriptionX := 400, DescriptionY := 1480,
DescriptionWidth := 0)
      END_STOP
      STEP  (ID := 22, X := 420, Y := 360, Operand :=
First_Step, HideDescription := Yes,
            DescriptionX := 478, DescriptionY := 345,
DescriptionWidth := 0,
            InitialStep := Yes, PresetUsesExpression := No,
LimitHighUsesExpression := No,
            LimitLowUsesExpression := No, ShowActions := Yes)
            ACTION  (ID := 24, Operand := First_Action,
Qualifier := L, IsBoolean := No,
            PresetUsesExpression := No, IndicatorTag :=
Watch_Tag[3].PRE)
            BODY  (LanguageType := ST)
            '
            END_BODY
            END_ACTION
            ACTION  (ID := 25, Operand := C_Array_Action[3],
Qualifier := SL,
            IsBoolean := No, PresetUsesExpression := No,
            IndicatorTag := C_Produced_IndicatorArray[1])
            BODY  (LanguageType := ST)
            '
            END_BODY
            END_ACTION
            ACTION  (ID := 26, Operand :=
UDT_Elem.Action_Member, Qualifier := D,
            IsBoolean := No, PresetUsesExpression := No,
IndicatorTag := "")
            BODY  (LanguageType := ST)
```

```
                '
                END_BODY
                END_ACTION
                ACTION  (ID := 27, Operand := Action_000,
Qualifier := R, IsBoolean := No,
                PresetUsesExpression := No, IndicatorTag := "")
                BODY  (LanguageType := ST)
                '
                END_BODY
                END_ACTION

                ACTION  (ID := 28, Operand := Action_001,
Qualifier := N, IsBoolean := No,
                PresetUsesExpression := No, IndicatorTag :=
Aliased_Indicator)
                BODY  (LanguageType := ST)
                '
                END_BODY
                END_ACTION
                ACTION  (ID := 29, Operand := Action_002,
Qualifier := DS, IsBoolean := Yes,
                PresetUsesExpression := No, IndicatorTag := "")
                END_ACTION
                ACTION  (ID := 30, Operand := ConsumedTag_Action,
Qualifier := P0,
                IsBoolean := No, PresetUsesExpression := No,
                IndicatorTag := ConsumedTag_Indicator)
                BODY  (LanguageType := ST)
                '
                END_BODY
                END_ACTION
        END_STEP
        STEP  (ID := 31, X := 120, Y := 880, Operand :=
"C_Array_Step[0,1,2]",
                HideDescription := Yes, DescriptionX := 179,
DescriptionY := 865,
                DescriptionWidth := 0, InitialStep := No,
PresetUsesExpression := No,
                LimitHighUsesExpression := No,
LimitLowUsesExpression := No, ShowActions := Yes)
        END_STEP
        TRANSITION  (ID := 33, X := 460, Y := 1000, Operand :=
NoTag_Tran,
                HideDescription := Yes, DescriptionX := 495,
DescriptionY := 985,
                DescriptionWidth := 0)
                CONDITION  (LanguageType := ST)
                'TempTag > 0
                END_CONDITION
        END_TRANSITION
        STEP  (ID := 35, X := 120, Y := 1080, Operand :=
UDT_Elem.Step_Member,
```

```
            HideDescription := Yes, DescriptionX := 199,
DescriptionY := 1065,
            DescriptionWidth := 0, InitialStep := No,
PresetUsesExpression := No,
            LimitHighUsesExpression := No,
LimitLowUsesExpression := No, ShowActions := Yes)
      END_STEP
      STEP  (ID := 37, X := 720, Y := 880, Operand :=
Step_001, HideDescription := No,
            DescriptionX := 760, DescriptionY := 940,
DescriptionWidth := 0,
            InitialStep := No, PresetUsesExpression := No,
LimitHighUsesExpression := No,
            LimitLowUsesExpression := No, ShowActions := Yes)
      END_STEP
      BRANCH  (ID := 39, Y := 1220, BranchType := Selection,
BranchFlow := Converge)
            LEG  (ID := 40)
            END_LEG
            LEG  (ID := 41)
            END_LEG
      END_BRANCH


      STEP  (ID := 42, X := 280, Y := 1260, Operand :=
Step_000, HideDescription := No,
            DescriptionX := 360, DescriptionY := 1240,
DescriptionWidth := 0,
            InitialStep := No, PresetUsesExpression := No,
LimitHighUsesExpression := No,
            LimitLowUsesExpression := No, ShowActions := Yes)
      END_STEP
      STEP  (ID := 44, X := 460, Y := 1080, Operand :=
ConsumedTag_Step,
            HideDescription := Yes, DescriptionX := 514,
DescriptionY := 1065,
            DescriptionWidth := 0, InitialStep := No,
PresetUsesExpression := No,
            LimitHighUsesExpression := No,
LimitLowUsesExpression := No, ShowActions := Yes)
      END_STEP
      TRANSITION  (ID := 46, X := 120, Y := 1160, Operand :=
UDT_Elem.Tran_Member,
            HideDescription := Yes, DescriptionX := 155,
DescriptionY := 1145,
            DescriptionWidth := 0)
            CONDITION  (LanguageType := ST)
            'TempTag > 0
            END_CONDITION
      END_TRANSITION
      DIRECTED_LINK  (FromElementID := 46, ToElementID := 41,
ShowLink := True)
      END_DIRECTED_LINK
      DIRECTED_LINK  (FromElementID := 15, ToElementID := 33,
```

```
ShowLink := True)
       END_DIRECTED_LINK
       DIRECTED_LINK  (FromElementID := 35, TToElementID := 46,
ShowLink := True)
       END_DIRECTED_LINK
       DIRECTED_LINK  (FromElementID := 3, ToElementID := 37,
ShowLink := True)
       END_DIRECTED_LINK
       DIRECTED_LINK  (FromElementID := 5, ToElementID := 31,
ShowLink := True)
       END_DIRECTED_LINK
       DIRECTED_LINK  (FromElementID := 6, ToElementID := 2,
ShowLink := True)
       END_DIRECTED_LINK
       DIRECTED_LINK  (FromElementID := 22, ToElementID := 6,
ShowLink := True)
       END_DIRECTED_LINK
       DIRECTED_LINK  (FromElementID := 16, ToElementID := 0,
ShowLink := True)
       END_DIRECTED_LINK
       DIRECTED_LINK  (FromElementID := 44, ToElementID := 12,
ShowLink := True)
       END_DIRECTED_LINK
       DIRECTED_LINK  (FromElementID := 33, ToElementID := 44,
ShowLink := True)
       END_DIRECTED_LINK
       DIRECTED_LINK  (FromElementID := 17, ToElementID := 10,
ShowLink := True)
       END_DIRECTED_LINK
       DIRECTED_LINK  (FromElementID := 42, ToElementID := 19,
ShowLink := True)
       END_DIRECTED_LINK
       DIRECTED_LINK  (FromElementID := 37, ToElementID := 18,
ShowLink := True)
       END_DIRECTED_LINK
       DIRECTED_LINK  (FromElementID := 4, ToElementID := 8,
ShowLink := True)
       END_DIRECTED_LINK


       DIRECTED_LINK  (FromElementID := 39, ToElementID := 42,
ShowLink := True)
       END_DIRECTED_LINK
       DIRECTED_LINK  (FromElementID := 10, ToElementID := 20,
ShowLink := True)
       END_DIRECTED_LINK
       DIRECTED_LINK  (FromElementID := 0, ToElementID := 35,
ShowLink := True)
       END_DIRECTED_LINK
       DIRECTED_LINK  (FromElementID := 31, ToElementID := 14,
ShowLink := True)
       END_DIRECTED_LINK
       DIRECTED_LINK  (FromElementID := 12, ToElementID := 40,
ShowLink := True)
```

```
        END_DIRECTED_LINK
        TEXT_BOX  (ID := 48, X := 260, Y := 1380, Width := 0,
              Text := "Simultaneous Branch Converge Text Box")
        END_TEXT_BOX
        ATTACHMENT  (FromElementID := 48, ToElementID := 17)
        END_ATTACHMENT
END_SFC_ROUTINE
```

# Define a structured text routine

## Introduction

This chapter explains how to enter structured text logic in a complete import/export file.

## Structured text routine

These examples show the structured text routine structure.

### L5X structured text structure

```
<Routines>
      <Routine [Routine_Attributes]>
            <Description>
                  <![CDATA[ text ]]>
            </Description>
            <STContent [StContent_Attributes]>
                  <Line Number="number">
                        <![CDATA[ structured_text ]]>
                  </Line>
            </STContent>
      </Routine>
</Routines>
```

### L5K structured text ST_ROUTINE structure

```
ST_ROUTINE <routine_name> [(Description := "text")]
      ‘(*comment_text*)
      ‘<statements>;
END_ST_ROUTINE;
```

## Structured Text routine elements

| L5X Item | L5K Item | Description |
|---|---|---|
| N/A | *routine_name* | The name of the routine.<br><br>In L5X, use a Name attribute on the <Routine> element. |
| Description | Description | User information about the routine. |
| STContent | *statements* | Structured text logic. |
| N/A | *comment_text* | Comment text within the structured text logic. |
| EncryptionInfo | ENCRYPTION_INFO | Details of the license-based Source Protection for the lockable object. Only exists for protected routines exported in plain text. |
| EncryptedContent | N/A | Source Protected and locked routine content. Only exists for locked routines exported in plain text. |
| EncryptedSegments | N/A | Locked logic for the routine. Only exists for locked routines exported in plain text. |

## ST_Routine attributes

| Attribute | Description |
|---|---|
| Name | L5X only. Specify the name of the routine.<br><br>In L5K, the name is an element of the statement. |
| Type | L5X only. Specify ST.<br><br>In L5K, the type of routine is part of the routine statement. |
| OnlineEditType | L5X only. Specify the online edit logic type. The options include **Original**, **PendingEdits**, or **TestEdits**. This attribute is not specified if there are no edits.<br><br>In L5X, this attribute is on the <STContent> element. |
| PermissionSet | Name of the set of permissions, configured in FactoryTalk Security, to apply to this object. |
| TrackingGroups | The group of tracked objects to which this item belongs. Components can be marked for tracking to determine whether they have been changed. Version 30 of the Logix Designer application supports only one tracking group. |

## Structured text logic

Enter the structured text logic within a routine component in an import/export file. Each line of structured text must begin with a single quote (‘).

Structured text is not case sensitive. Structured text can contain these elements.

| Term | Definition | | Examples |
|---|---|---|---|
| Assignment | Use an assignment statement to assign values to tags.<br>The := operator is the assignment operator.<br>Terminate the assignment with a semi colon ";". | | `tag := expression;` |
| Expression | An expression is part of a complete assignment or construct statement. An expression evaluates to a number (numerical expression) or to a true or false state (BOOL expression).<br><br>An expression contains these elements. | | |
| | Tags | A named area of the memory where data is stored (BOOL, SINT, INT, DINT, REAL, string). | `value1` |
| | Immediates | A constant value. | `4` |
| | Operators | A symbol or mnemonic that specifies an operation within an expression. | `tag1 + tag2`<br>`tag1 >= value1` |
| | Functions | When executed, a function yields one value. Use parentheses to contain the operand of a function.<br><br>Functions can be used only in expressions. | `function(tag1)` |

| Term | Definition | Examples |
|---|---|---|
| Instruction | An instruction is a standalone statement.<br>An instruction uses parenthesis to contain its operands.<br>Depending on the instruction, there can be zero, one, or multiple operands.<br>When executed, an instruction yields one or more values that are part of a data structure.<br>Terminate the instruction with a semi colon ";".<br><br>Instructions cannot be used in expressions. | `instruction();`<br><br>`instruction(operand);`<br><br>`instruction(operand1, operand2,operand3);` |
| Construct | A conditional statement used to trigger structured text code.<br>Terminate the construct with a semi colon ";". | `IF...THEN`<br>`CASE`<br>`FOR...DO`<br>`WHILE...DO`<br>`REPEAT...UNTIL`<br>`EXIT` |
| Comment | Text that explains or clarifies what a section of structured text does.<br>• Use comments to make it easier to interpret the structured text.<br>• Comments do not affect the execution of the structured text.<br>• Comments can appear anywhere in structured text. | `//comment`<br><br>`(*start of comment... end of comment*)`<br><br>`/*start of comment... end of comment*/` |

## Export structured text logic while editing online

If you export structured text logic that contains online edits, the export file exports LOGIC blocks (in L5K format) or additional <STContent> elements (in L5X format) to indicate the original test edits and pending edits states. If there are no online edits, you will not see these LOGIC blocks or additional <STContent> elements.

| Item | Identifies |
|---|---|
| Online_Edit_Type | When you export the logic:<br>• If online edits exist, there is a LOGIC block for Online_Edit_Type := Orig and the appropriate LOGIC block for the existing edits. Online_Edit_Type : = Pend indicates pending edits. Online_Edit_Type := Test indicates test edits.<br>• If there are no online edits when you export the logic, there are no LOGIC blocks and the main components in the routine are structured text statements. |

# Examples

**L5X example: test edits and pending edits exist**

```
- <Routine Name="STRoutine" Type="ST">
  - <STContent OnlineEditType="Original">
      <!--  ST content inserted here - see format descibed above   -->
    </STContent>
  - <STContent OnlineEditType="PendingEdits">
      <!--  ST content inserted here - see format descibed above   -->
    </STContent>
  - <STContent OnlineEditType="TestEdits">
      <!--  ST content inserted here - see format descibed above   -->
    </STContent>
  </Routine>
```

**L5K Example 1: Test edits and pending edits exist**

```
ST_ROUTINE MySTRoutine
       LOGIC (Online_Edit_Type := Orig)
              (* structured text logic here *)
       END_LOGIC

       LOGIC (Online_Edit_Type := Test)
              (* structured text logic here *)
       END_LOGIC

       LOGIC (Online_Edit_Type := Pend)
              (* structured text logix here *)
       END_LOGIC
END_ST_ROUTINE
```

**L5K Example 2: Only pending edits exist**

```
ST_ROUTINE MySTRoutine
       LOGIC (Online_Edit_Type := Orig)
              (* structured text logic here *)
       END_LOGIC

       LOGIC (Online_Edit_Type := Pend)
              (* structured text logic here *)
       END_LOGIC
END_ST_ROUTINE
```

#### L5X structured text routine example

```
- <Routine Name="ST_Gear_Change" Type="ST">
  - <STContent>
    - <Line Number="0">
        <![CDATA[     (*** This program demonstrates how ST Language
        programs work. ***)   ]]>
      </Line>
    - <Line Number="1">
        <![CDATA[  ]]>
      </Line>
    - <Line Number="2">
        <![CDATA[ If State = 0 then  ]]>
      </Line>
    - <Line Number="3">
        <![CDATA[     MSO (axis0, axis0_MSO);                (* Turn
        servos on and set tags to initial values *)  ]]>
      </Line>
    - <Line Number="4">
        <![CDATA[     MSO (axis1, axis1_MSO);  ]]>
      </Line>
    - <Line Number="5">
        <![CDATA[     gear_ratio [:=] 0;  ]]>
      </Line>
    - <Line Number="6">
        <![CDATA[     State [:=] 1;  ]]>
      </Line>
    - <Line Number="7">
        <![CDATA[ end_if;  ]]>
      </Line>
    </STContent>
  </Routine>
```

#### L5K structured text ST_ROUTINE example

This is an example of an exported structured text routine.

```
ST_ROUTINE <routine_name>
(*----------- Sample of ST code-----------------------------*)
             'IF (myInteger = 12) THEN
             '      myInteger := ((5 * myInputInteger1) + (7 *
myInteger2)) - 71;
             '             WHILE (myTmpVar >= 0) DO
             '                    myInteger := myInteger + 3;
             '                    myTmpVar := myTmpVar - 1;
             '             END_WHILE;
             'END_IF;
ND_ST_ROUTINE
```

## Structured text

These tables list each structured text instruction and function. For more details, see these reference manuals.

| Instruction Type | Resource |
|---|---|
| Basic, sequential instruction | Logix5000 Controllers General Instructions Set Reference Manual, publication 1756-RM003. |
| Process control or drives instruction | Logix5000 Controllers Advanced Process Control and Drives Instruction Set Reference Manual, publication1756-RM006. |
| Motion instruction | Logix5000 Controllers Motion Instructions Reference Manual, publication MOTION-RM002. |

**Structured Text Instructions**

| Instruction | Neutral Text Format |
|---|---|
| ABL | `ABL(Channel,SerialPortControl);` |
| ABS | `dest := ABS(source);` |
| ACB | `ACB(Channel,SerialPortControl);` |
| ACL | `ACL(Channel,ClearSerialPortRead,ClearSerialPortWrite);` |
| ACOS | `dest := ACOS(source);` |
| ADD | `dest := sourceA + sourceB;` |
| AHL | `AHL(Channel,ANDMask,ORMask,SerialPortControl);` |
| ALM | `ALM(ALM_tag);` |
| ALMA | `ALMA (ALMA_tag,In,ProgAckAll,ProgramDisable,ProgEnable);` |
| ALMD | `ALMD (ALMD_tag,In,ProgAck,ProgReset,ProgDisable,ProgEnable);` |
| AND | `dest := sourceA & sourceB;`<br>`dest := sourceA AND sourceB;` |
| ARD | `ARD(Channel,Destination,SerialPortControl);` |
| ARL | `ARL(Channel,Destination,SerialPortControl);` |
| ASIN | `dest := ASIN(source);` |
| ATAN | `dest := ATAN(source);` |
| AWA | `AWA(Channel,Source,SerialPortControl);` |
| AWT | `AWT(Channel,Source,SerialPortControl);` |
| BAND | `IF operandA AND operandB THEN`<br>`  <statement>;`<br>`ENDIF;` |
| BNOT | `IF NOT operand THEN`<br>`  <statements>;`<br>`ENDIF;` |
| BOR | `IF operandA OR operandB THEN`<br>`  <statements>;`<br>`ENDIF;` |
| BTDT | `BTD(BTDT_tag);` |

| Instruction | Neutral Text Format |
|---|---|
| BXOR | IF *operandA* XOR *operandB* THEN<br>  *<statements>*;<br>ENDIF; |
| CASE...OF | CASE *numeric_expression* OF<br>  selector1: statement;<br>  selectorN: statement;<br>ELSE<br>  statement;<br>END_CASE; |
| CLR | *dest* := 0; |
| CONCAT | CONCAT(*SourceA,SourceB,Dest*) |
| COP | COP(*Source,Dest,Length*); |
| COS | *dest* := COS(*source*); |
| CPS | CPS(*Source,Dest,Length*) |
| CTUD | CTUD(*CTUD_tag*); |
| D2SD | D2SD(*D2SD_tag*); |
| D3SD | D3SD(*D3SD_tag*); |
| DEDT | DEDT(*DEDT_tag,storage*); |
| DEG | *dest* := DEG(*source*); |
| DELETE | DELETE(*Source,Qty,Start,Dest*); |
| DERV | DERV(*DERV_tag*); |
| DFF | DFF(*DFF_tag*); |
| DIV | dest := *sourceA / sourceB*; |
| DTOS | DTOS(*Source,Dest*); |
| EOT | EOT(*DataBit*); |
| EQU | IF *sourceA* = source*B* THEN<br>  *<statements>*;<br>ENDIF; |
| ESEL | ESEL(*ESEL_tag*); |
| EVENT | EVENT(*task*); |
| FGEN | FGEN(*FGEN_tag,X1,Y1,X2,Y2*); |
| FIND | FIND(*Source,Search,Start,Result*) |
| FOR...DO | FOR *count*:= *initial_value* TO *final_value* BY *increment* DO<br>  <statement>;<br>END_FOR; |

| Instruction | Neutral Text Format |
|---|---|
| GEQ | IF *sourceA* >= source*B* THEN<br>  <*statements*>;<br>ENDIF; |
| GRT | IF *sourceA* > source*B* THEN<br>  <*statements*>;<br>ENDIF; |
| GSV | GSV(*ClassName,InstanceName,AttributeName,Dest*); |
| HLL | HLL(*HLL_tag*); |
| HPF | HPF(*HPF_tag*); |
| IF...THEN | IF *bool_expression* THEN<br>  <*statement*>;<br>END_IF; |
| INSERT | INSERT(*SourceA,SourceB,Start,Dest*); |
| INTG | INTG(*INTG_tag*); |
| IOT | IOT(*output_tag*); |
| JKFF | JKFF(*JKFF_tag*); |
| JSR | JSR(*RoutineName,InputCount,InputPar,ReturnPar*); |
| LDL2 | LDL2(*LDL2_tag*); |
| LDLG | LDLG(*LDLG_tag*); |
| LEQ | IF *sourceA* <= source*B* THEN<br>  <*statements*>;<br>ENDIF; |
| LES | IF *sourceA* < source*B* THEN<br>  <*statements*>;<br>ENDIF; |
| LN | *dest* := LN(*source*); |
| LOG | *dest* := LOG(*source*); |
| LOWER | LOWER(*Source,Dest);* |
| LPF | LPF(*LPF_tag*); |
| MAAT | MAAT(*Axis,MotionControl*); |
| MAFR | MAFR(*Axis,MotionControl*); |
| MAG | MAG(Slave*Axis,MasterAxis,MotionControl,Direction,Ratio,SlaveCounts,MasterCounts,MasterReference,RatioFormat,Clutch,AccelRate,AccelUnits*); |
| MAH | MAH(*Axis,MotionControl*); |
| MAHD | MAHD(*Axis,MotionControl,DiagnosticTest,ObservedDirection*); |
| MAJ | MAJ(*Axis,MotionControl,Direction,Speed,SpeedUnits,AccelRate,AccelUnits,DecelRate,DecelUnits,Profile,Merge,MergeSpeed*); |
| MAM | MAM(*Axis,MotionControl,MoveType,Position,Speed,SpeedUnits,AccelRate,AccelUnits,DecelRate,DecelUnits,Profile,Merge,MergeSpeed*); |

| Instruction | Neutral Text Format |
|---|---|
| MAOC | MAOC(*Axis,ExecutionTarget,MotionControl,Output,Input,OutputCam, CamStartPosition,CamEndPosition,OutputCompensation,ExecutionMode, ExecutionSchedule,AxisArmPosition,CamArmPosition,Reference*); |
| MAPC | MAPC(*SlaveAxis*,*MasterAxis*,*MotionControl*,*Direction*,*CamProfile*, *SlaveScaling*,*MasterScaling*,*ExecutionMode*,*ExecutionSchedule*, *MasterLockPosition*,*CamLockPosition*,*MasterReference*,*MasterDirection*); |
| MAR | MAR(*Axis,MotionControl,TriggerCondition,WindowedRegistration, MinimumPosition,MaximumPosition*); |
| MAS | MAS(*Axis,MotionControl,StopType,ChangeDecel,DecelRate,DecelUnits*); |
| MASD | MASD(*Axis,MotionControl*); |
| MASR | MASR(*Axis,MotionControl*); |
| MATC | MATC(*Axis*,*MotionControl*,*Direction*,*CamProfile*,*DistanceScaling*, *TimeScaling*,*ExecutionMode*,*ExecutionSchedule*); |
| MAVE | MAVE(*MAVE_tag,storage,weight*); |
| MAW | MAW(*Axis,MotionControl,TriggerCondition,Position*); |
| MAXC | MAXC(*MAXC_tag*); |
| MCCD | MCCD(*Coordinate_system,MotionControl,MotionType,ChangeSpeed,Speed, SpeedUnits,ChangeAccel,AccelRate,AccelUnits,ChangeDecel,DecelRate, DecelUnits,Scope*); |
| MCCM | MCCM(*CoordinateSystem,MotionControl,MoveType,Position,CircleType, Via/Center/Radius,Direction,Speed,SpeedUnits,AccelRate,AccelUnits, DecelRate,DecelUnits,Profile,TerminationType,Merge,MergeSpeed*); |
| MCCP | MCCP(*MotionControl*,*Cam*,*Length*,*StartSlope*,*EndSlope*,*CamProfile*); |
| MCD | MCD(*Axis,MotionControl,MotionType,ChangeSpeed,Speed,ChangeAccel, AccelRate,ChangeDecel,DecelRate,SpeedUnits,AccelUnits,DecelUnits*); |
| MCLM | MCLM(*CoordinateSystem,MotionControl,MoveType,Position,Speed,SpeedUnits, AccelRate,AccelUnits,DecelRate,DecelUnits,Profile,TerminationType,Merge, MergeSpeed*); |
| MCS | MCS(*CoordinateSystem*,*MotionControl*,*StopType*,*ChangeDecel*,*DecelRate*, *DecelUnits*); |
| MCSD | MCSD(*CoordinateSystem,MotionControl*); |
| MCSR | MCSR(*CoordinateSystem,MotionControl*); |
| MCSV | MCSV(*MotionControl,CamProfile,MasterValue,SlaveValue,SlopeValue, SlopeDerivative*); |
| MCT | MCT(*SourceSystem,TargetSystem,MotionControl,Orientation,Translation*); |
| MCTP | MCTP(*SourceSystem,TargetSystem,MotionControl,Orientation,Translation, TransformDirection,ReferencePosition,TransformPosition*); |
| MDF | MDF(*Axis,MotionControl*); |
| MDO | MDO(*Axis,MotionControl,DriveOutput,DriveUnits*); |
| MDOC | MDOC(*Axis,ExecutionTarget,MotionControl,DisarmType*); |
| MDR | MDR(*Axis,MotionControl*); |
| MDW | MDW(*Axis,MotionControl*); |

| Instruction | Neutral Text Format |
|---|---|
| MEQ | IF (Source AND Mask) = (Compare AND Mask) THEN<br><*statements*>;<br>END_IF; |
| MGS | MGS(*Group,MotionControl,StopMode*); |
| MGSD | MGSD(*Group,MotionControl*); |
| MGSP | MGSP(*Group,MotionControl*); |
| MGSR | MGSR(*Group,MotionControl*); |
| MID | MID(*Source,Qty,Start,Dest*); |
| MINC | MINC(*MINC_tag*); |
| MOD | *dest* := *sourceA* MOD *sourceB*; |
| MRAT | MRAT(*Axis,MotionControl*); |
| MRHD | MRHD(*Axis,MotionControl,DiagnosticTest*); |
| MRP | MRP(A*xis,MotionControl,Type,PositionSelect,Position*); |
| MSF | MSF(*Axis,MotionControl*); |
| MSG | MSG(*MessageControl*); |
| MSO | MSO(*Axis,MotionControl*); |
| MUL | *dest* := *sourceA* * *sourceB*; |
| MVMT | MVMT(*MVMT_tag*); |
| NEG | *dest* := -*source*; |
| NEQ | IF *sourceA* <> *sourceB* THEN<br>  <*statements*>;<br>END_IF; |
| NOT | IF NOT *source* THEN<br>  <*statements*>;<br>END_IF; |
| OR | *dest* := *sourceA* OR *sourceB* |
| OSFI | OSFI(*OSFI_tag*); |
| OSRI | OSRI(*OSRI_tag*); |
| OTE | *data_bit* [:=] *BOOL_expression*; |
| OTL | IF *BOOL_expression* THEN<br>  *data_bit* := 1;<br>END_IF; |
| OTU | IF *BOOL_expression* THEN<br>  *data_bit* := 0;<br>END_IF; |
| PATT | PATT(P*haseName,Result*); |
| PCLF | PCLF(*PhaseName*); |
| PCMD | PCMD(*PhaseName,Command,Result*); |
| PDET | PDET(*PhaseName*); |

| Instruction | Neutral Text Format |
|---|---|
| PFL | PFL(*Source*); |
| PI | PI(*PI_tag*); |
| PID | PID(PID,*ProcessVariable*,*Tieback*,*ControlVariable*,*PIDMasterLoop*,*InholdBit*, *InholdValue);* |
| PIDE | PIDE(*PIDE_tag*); |
| PMUL | PMUL(*PMUL_tag*); |
| POSP | POSP(*POSP_tag*); |
| POVR | POVR(*PhaseName*,*Command*,*Result);* |
| PPD | PPD(); |
| PRNP | PRNP(); |
| PSC | PSC(); |
| PXRQ | PXRQ(*PhaseInstruction*,*ExternalRequest*,*DataValue*); |
| RAD | *dest* := RAD(*source*); |
| REPEAT...UNTIL | REPEAT<br><statement>;<br>UNTIL bool_expression<br>END_REPEAT; |
| RESD | RESD(*RESD_tag*); |
| RET | RET(*ReturnPar)*; |
| RLIM | RLIM(*RLIM_tag*); |
| RMPS | RMPS(*RMPS_tag*,*RampValue*,*SoakValue*,*SoakTime*); |
| RTOR | RTOR(*RTOR_tag*); |
| RTOS | RTOS(*Source*,*Dest*) |
| SBR | SBR(*InputPar*); |
| SCRV | SCRV(*SCRV_tag*); |
| SETD | SETD(*SETD_tag*); |
| SFP | SFP(SFCRoutineName,TargetState); |
| SFR | SFR(SFCRoutineName,StepName); |
| SIN | dest := SIN(*source*); |
| SIZE | SIZE(*Souce*,*Dimensiontovary*,*Size*); |
| SNEG | SNEG(*SNEG_tag*); |
| SOC | SOC(*SOC_tag*); |
| SQRT | *dest* := SQRT(*source*); |
| SRT | SRT(*Array*,*Dimtovary*,*Control*); |
| SRTP | SRTP(*SRTP_tag*); |
| SSUM | SSUM(*SSUM_tag*); |

| Instruction | Neutral Text Format |
|---|---|
| SSV | SSV(*ClassName,InstanceName,AttributeName,Source*); |
| STOD | STOD(*Source,Dest*) |
| STOR | STOR(*Source,Dest*) |
| SUB | *dest* := *sourceA - sourceB*; |
| SWPB | SWPB(*Source,OrderMode,Dest*); |
| TAN | *dest* := TAN(*source*); |
| TOFR | TOFR(*TOFR_tag*); |
| TONR | TONR(*TONR_tag*); |
| TOT | TOT(*TOT_tag*); |
| TRUNC | *dest* := TRUNC(*source*); |
| UID | UID(); |
| UIE | UIE(); |
| UPDN | UPDN(*UPDN_tag*); |
| UPPER | UPPER(*Source,Destination);* |
| WHILE...DO | WHILE *bool_expression* DO<br><br><statement>;<br><br>END_WHILE; |
| XIC | IF *data_bit* THEN<br><*statement*>;<br>END_IF; |
| XIO | IF NOT *data_bit* THEN<br><*statement*>;<br>END_IF; |
| XOR | *dest* := *sourceA* XOR *sourceB*; |
| XPY | *dest* := *sourceX* XPY *sourceY*; |

# Define an Equipment Sequence routine

## Introduction

This chapter explains how to define an Equipment Sequence routine in a complete import/export file.

## Equipment Sequence Routine

The following examples show Equipment Sequence structure.

### L5X Equipment Sequence structure

```
<Routines>
      <Routine [Routine_Attributes]>
            <Description>
                  <![CDATA[ text ]]>
            </Description>
            <SEQContent [SEQContent_Attributes]>
                  <Step [Step_Attributes]/>
                  <Transition [Transition_Attributes]/>
                  <Branch [Branch_Attributes]/>
                  <Stop [Stop_Attributes]/>
                  <DirectedLink
[DirectedLink_Attributes]/>
                  <TextBox [TextBox_Attributes]>
                        text
                  </TextBox>
                  <Attachment [Attachment_Attributes]/>
                  <TagConfigurations>
                        [TagConfiguration]
```

## L5K Equipment Sequence ESQ_ROUTINE structure

```
                    </TagConfigurations/>
                </SEQContent>
            </Routine>
        </Routines>
        ESQ_ROUTINE <routine_name> [(Routine_Attributes,
                SFC_Attributes)]
            <STEP declaration>
            <TRANSITION declaration>
            <BRANCH declaration>
            <STOP declaration>
            <DIRECTED_LINK declaration>
            <TEXT_BOX declaration)
            <ATTACHMENT declaration>
            <TAG_CONFIGURATION declaration>
        END_ESQ_ROUTINE
```

## Equipment Sequence elements

| L5X Item | L5K Item | Identifies |
|---|---|---|
| N/A | routine_name | The name of the routine.<br><br>In L5X, use a Name attribute on the <Routine> element. |
| Description | Description | User information about the routine. |
| SEQContent | N/A | Equipment sequence logic. |

## ESQ_Routine attributes

| Attribute | Description |
|---|---|
| Name | L5X only. Specify the name of the routine.<br><br>For L5K, the name is an element of the statement. |
| Type | L5X only. Specify "Sequence".<br><br>In L5K, the type of routine is part of the routine statement. |

| Attribute | Description |
|---|---|
| SheetSize | The size of the Equipment Sequence. Select one of these options.<br>• Letter (8.5x11in)<br>• Legal (8.5x14in)<br>• Tabloid (11x17in)<br>• A4 (210x297mm)<br>• A3 (297x420mm)<br><br>In L5X, this attribute is on the <SEQContent> element. |
| SheetOrientation | The orientation of the Equipment Sequence sheet. Select **Portrait** or **Landscape**.<br><br>In L5X, this attribute is on the <SEQContent> element. |
| PermissionSet | Name of the set of permissions, configured in FactoryTalk Security, to apply to this object. |

# Steps

These examples show the step structure.

## L5X step structure

```
<Step [Step_Attributes] />
```

## L5K STEP structure

```
STEP (Step_Attributes)

END_STEP
```

## Step attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| ID | ID | The step identifier. This ID uniquely identifies this step from all other blocks. Type an unsigned, 32-bit integer value. |
| X | X | X-coordinate on internal grid. Type an unsigned, 32-bit integer value. |
| Y | Y | Y-coordinate on internal grid. Type an unsigned, 32-bit integer value. |
| Operand | Operand | The step tag. Type a tag of datatype ESQ_STEP. The import process uses this tag name to name the step. |
| HideDesc | HideDescription | Whether or not to hide the step description. Type **Yes** or **No**. |
| DescX | DescriptionX | X-coordinate on internal grid of the description box. Type an unsigned, 32-bit integer value. |
| DescY | DescriptionY | Y-coordinate on internal grid of the description box. Type unsigned, 32-bit integer value. |
| DescWidth | DescriptionWidth | This attribute is not currently used; it is there for future use. Type **0**. |

| L5X Item | L5K Item | Identifies |
|---|---|---|
| InitialStep | InitialStep | Whether this step is the initial step of the routine. Type **Yes** or **No**.<br>If you have multiple steps identified as the initial step, which is incorrect syntax, the import process designates the last initial step it encounters as the initial step and removes the initial step indicators from any other steps. |
| NoPhaseStep | NoPhaseStep | Whether this step is a place holder step, a step which is configured not to start any equipment phase. PhaseName attributes are ignored if NoPhaseStep is set to **Yes** in L5K, **true** in L5X.<br><br>L5K: **Yes**/**No**, L5X: **true**/**false**. |
| PhaseName | PhaseName | The name of the Phase associated with the step. Can be an empty string. |
| TransferOfControlSource | TransferOfControlSource | Specifies whether this step will transfer control of its equipment phase to an immediately following step without stopping and resetting the phase.<br><br>L5K: **Yes**/**No**, L5X: **true**/**false**. |
| TransferOfControlTarget | TransferOfControlTarget | Specifies whether this step will accept the transfer control of its equipment phase from an immediately preceding step without starting the phase. The phase logic is expected to be executing a state routine. The executing routine is notified of the transfer of control and can request new input parameters, if programmed to do so, from the target step.<br><br>L5K: **Yes**/**No**, L5X: **true**/**false**. |

# Transitions

These examples show transition structure.

## L5X transition structure

```
<Transition [Transition_Attributes]>
      <Condition>
            logic
      </Condition>
</Transition>
```

## L5K TRANSITION structure

```
TRANSITION (Transition_Attributes)
      <CONDITION declaration>
END_TRANSITION
```

## Transition elements

| L5X Item | L5K Item | Identifies |
|---|---|---|
| Condition | CONDITION | The condition to evaluate for the transition. |

## Transition attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| ID | ID | The step identifier. This ID uniquely identifies this step from all other blocks. Type an unsigned, 32-bit integer value. |
| X | X | X-coordinate on internal grid. Type an unsigned, 32-bit integer value. |
| Y | Y | Y-coordinate on internal grid. Type an unsigned, 32-bit integer value. |
| Operand | Operand | The step tag. Type a tag of datatype SFC_STEP. The import process uses this tag name to name the step. |
| HideDesc | HideDescription | Whether or not to hide the step description. Type **Yes** or **No**. |
| DescX | DescriptionX | X-coordinate on internal grid of the description box. Type an unsigned, 32-bit integer value. |
| DescY | DescriptionY | Y-coordinate on internal grid of the description box. Type unsigned, 32-bit integer value. |
| DescWidth | DescriptionWidth | This attribute is not currently used; it is there for future use. Type **0**. |
| Force | Force | The transition is forced. Type **true** for forced true (set) or type **false** for forced false (cleared). If the transition is not forced, do not enter this attribute. |

# Stops

These examples show stop structure.

## L5X stop structure

```
<Stop [StopAttributes]/>
```

## L5K STOP structure

```
STOP (Stop_Attributes)
END_STOP
```

## Stop attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| ID | ID | The stop identifier. This ID uniquely identifies this stop from all other blocks. Type an unsigned, 32-bit integer value. |
| X | X | X-coordinate on internal grid. Type an unsigned, 32-bit integer value. |
| Y | Y | Y-coordinate on internal grid. Type an unsigned, 32-bit integer value. |
| Operand | Operand | The stop tag. Type a tag of datatype SFC_STOP. The import process uses this tag name to name the stop. |
| HideDesc | HideDescription | Whether or not to hide the stop description. Type **Yes** or **No**. |
| DescX | DescriptionX | X-coordinate on internal grid of the description box. Type an unsigned, 32-bit integer value. |
| DescY | DescriptionY | Y-coordinate on internal grid of the description box. Type unsigned, 32-bit integer value. |
| DescWidth | DescriptionWidth | This attribute is not currently used; it is there for future use. Type **0**. |

## Branches

The branch blocks in an Equipment Sequence routine identify simultaneous or selection branches in the routine.

### L5X branch structure

```
<Branch [BranchAttributes]>
      <Leg [LegAttributes]/>
</Branch>
```

### L5K BRANCH structure

```
BRANCH (Branch_Attributes)
      LEG (Leg_Attributes)
      END_LEG
END_BRANCH
```

### Branch attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| ID | ID | The branch identifier. This ID uniquely identifies this branch from all other blocks. Type an unsigned, 32-bit integer value. |
| X | X | X-coordinate on internal grid. Type an unsigned, 32-bit integer value. |
| Y | Y | Y-coordinate on internal grid. Type an unsigned, 32-bit integer value. |
| BranchType | BranchType | The type of branch. Type **Simultaneous** or **Selection**. |
| BranchFlow | BranchFlow | The direction of the branch. Type **Converge** or **Diverge**. |
| Priority | Priority | Whether the priority of a divergent selection branch is defined by the user. This attribute applies only to divergent selection branches. Type **Default** or **UserDefined**. |

### Leg attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| ID | ID | The leg identifier. This ID uniquely identifies this leg from all other blocks. Type an unsigned, 32-bit integer value. |
| Force | Force | Whether the leg is forced or not. You can force only a leg in a simultaneous branch. Omit this attribute, for no forces, or type **false** to force the leg false. |

## Directed links

The directed link blocks in an Equipment Sequence routine identify the links between Equipment Sequence components.

### L5X DirectedLink structure

```
<DirectedLink  [DirectedLinkAttributes]/>
```

### L5K DIRECTED_LINK structure

```
DIRECTED_LINK (Directed_Link_Attributes)
END_DIRECTED_LINK
```

### Directed link attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| FromID | FromElementID | The ID of the object. Type an unsigned, 32-bit integer value. |
| ToID | ToElementID | The ID of the object that the FromID object is attached to. Type an unsigned, 32-bit integer value. |
| Show | ShowLink | Whether or not to show the link. Type **TRUE** or **FALSE**. |

### Directed link guidelines

Use these guidelines for directed links:

- All directed link blocks must come after all step, transition, stop, and branch blocks.
- A directed link links only one element to one other element.

## Attachments

The attachment blocks in an Equipment Sequence routine identify the attachments from text boxes to other Equipment Sequence elements.

### L5X attachment structure

```
<Attachment [Attachment_Attributes]/>
```

### L5K ATTACHMENT structure

```
ATTACHMENT (Attachment_Attributes)
END_ATTACHMENT
```

### Attachment attributes

| L5X Item | L5K Item | Identifies |
|---|---|---|
| FromID | FromElementID | The ID of the attached object. Type an unsigned, 32-bit integer value. |
| ToID | ToElementID | The ID of the object that the FromID object is attached to. Type an unsigned, 32-bit integer value. |

### Attachment guidelines

Use these guidelines for attachments:

- Use an attachment to link a text box to an Equipment Sequence element.
- All attachment blocks must come after all text box blocks.

## Tag configuration

The following examples show the tag configuration structure.

### L5X tag configuration structure

```
<TagConfigurations>
<TagConfiguration [Tag_Configuration_Attributes]>
      <Expression>
logic
      </Expression>
```

```
</TagConfiguration>
</TagConfigurations>
```

### L5K TAG_CONFIGURATION structure

```
TAG_CONFIGURATION (Step_Attributes)
      <EXPRESSION declaration>
END_TAG_CONFIGURATION
```

### Tag configuration elements

| L5X Item | L5K Item | Description |
|---|---|---|
| Expression | EXPRESSION | The expression to evaluate for the tag. |

## Expression component

The expression component uses a structured text expression to specify an expression to evaluate for the sequence tag.

### L5X expression structure

```
<Expression>
      <STContent>
            <Line Number="0">
                  <![CDATA[ structured_text; ]]>
            </Line>
      </STContent>
</Expression>
```

### L5K EXPRESSION structure

```
EXPRESSION (LanguageType := ST)
      ‘<structured_text>
END_EXPRESSION
```

**Tip:** Each line of L5K structured text begins with a single quote (‘).

# Define a task component

## Introduction

This chapter explains the overall structure of the task component.

## Task component

A task component defines a task in the controller project. The maximum number of tasks depends on the type of controller.

| Controller | Maximum Number of Tasks |
|---|---|
| ControlLogix | 32 |
| SoftLogix5800 | 32 |
| FlexLogix | 8 |
| CompactLogix<br>• 1768-L30x<br>• 1769-L33x<br>• 1769-L36x | <br>32<br>32<br>23 |
| DriveLogix | 8 |

## L5K TASK structure

```
TASK <task_name> [(Description := "text", Attributes)]
      <program_name>;
END_TASK
```

## L5X task structure

```
<Tasks>
      <Task [Task_Attributes]>
            <Description>
                  <![CDATA[ text ]]>
            </Description>
            <EventInfo [EventInfo_Attributes]>
            <ScheduledPrograms>
            <     ScheduledProgram Name="program_name" />
            </Programs>
      </Task>
</Tasks>
```

## Task elements

| L5X Item | L5K Item | Description |
|---|---|---|
| N/A | *task_name* | The name of the task.<br><br>In L5X, use a Name attribute on the <Task> element. |
| Description | Description | User information about the task. |
| EventInfo | N/A | Event information for an event task |
| ScheduledProgram | *program_name* | Each program within the task. |

## Task attributes

| Attribute | Description |
|---|---|
| Name | L5X only. Specify the name of the task.<br><br>In L5K, the name is an element of the statement. |
| Type | Specify whether the type of task is **Continuous**, **Periodic**, or **Event**. There can be only one continuous task. |
| Class | Specify the class of the task. This attribute applies only to safety controller projects. Type **Standard** or **Safety**. |
| Rate | If the task is a periodic task, specify how often to run the task (1.000...2,000,000.000 µs). |
| Priority | Specify the priority of a periodic task (1...15) |
| Watchdog | Type the watchdog timeout for the task (1.000...2,000,000.000 µs). |
| EventTrigger | Only used for event tasks.<br>Specify the trigger for the event task. Type **Axis Home**, **Axis Watch**, **Axis Registration 1**, **Axis Registration 2**, **Motion Group Execution**, **EVENT Instruction Only**, **Module Input Data State Change**, **Consumed Tag**, or **Windows Event**.<br><br>In L5X, this attribute is on the <EventInfo> element. |
| EventTag | Only used for event tasks with a Consumed Tag trigger or a Module Input Data State Change trigger.<br>Specify the tag to consume.<br><br>In L5X, this attribute is on the <EventInfo> element. |
| EnableTimeout | Type **Yes** to enable timeouts for the task. Otherwise type **No**.<br><br>In L5X, this attribute is on the <EventInfo> element. |
| DisableUpdateOutputs | Type **Yes** to disable updates to outputs while the task executes. Otherwise type **No**. The default for a periodic or continuous task is No. The default for an event task is **yes**. |
| InihibitTask | Type **Yes** to inhibit the task. Otherwise enter No. |

## Task guidelines

Observe these guidelines when defining a task:

- Tasks must be defined after programs and before controller objects.
- There is a maximum of 32 tasks.
- There is one continuous task only.
- A program can be scheduled under one task only.
- Scheduled programs must be defined (must exist).

## Examples

### L5X Task example

```
- <Task Name="MainTask" Type="PERIODIC" Rate="100" Priority="10" Watchdog="500"
    DisableUpdateOutputs="false" InhibitTask="false">
  - <ScheduledPrograms>
      <ScheduledProgram Name="Recipe_Ops" />
      <ScheduledProgram Name="Add_Egg_M2" />
      <ScheduledProgram Name="Add_Sugar_M2" />
      <ScheduledProgram Name="Add_Cream_M2" />
      <ScheduledProgram Name="Add_Milk_M2" />
      <ScheduledProgram Name="Agitate_M2" />
      <ScheduledProgram Name="Heat_M2" />
    </ScheduledPrograms>
  </Task>
```

### L5K TASK example

```
TASK joe (Type := Periodic,  Priority := 8, Rate := 10000)
       sue;
       betty;
END_TASK
```

You can define the task attributes (Type, Priority, Rate, and Watchdog) in any order. The list of programs scheduled for a task are listed in the task declarations block, as shown above. The programs are executed in the order they are specified.

### L5K Safety TASK Example

```
TASK SafetyTask (Type := PERIODIC,
       Class := Safety,
       Rate := 10,
       Priority := 10,
       Watchdog := 10,
       DisableUpdateOutputs := No,
       InhibitTask := No)
       SafetyProgram;
END_TASK
```

# Define a parameter connection component

## Introduction

This chapter explains the overall structure of the parameter connection component.

## Parameter connection component

A parameter connection component defines a connection between two program parameters, which allows you to share data between programs without using controller-scope tags.

## L5K PARAMETER_CONNECTION structure

```
PARAMETER_CONNECTION (EndPoint1:=<program.parameter_name>,
                      EndPoint2:=<program.parameter_name>)
END_PARAMETER_CONNECTION
```

## L5X ParameterConnection structure

```
<ParameterConnections>
      <ParameterConnection
EndPoint1="program_name.parameter_name",

EndPoint2="program_name.parameter_name" />
</ParameterConnections>
```

## Parameter connection attributes

| Attribute | Description |
|---|---|
| EndPoint1 | Specify the first end point of the parameter connection. |
| EndPoint2 | Specify the second end point of the parameter connection. |

## Parameter connection guidelines

Observe this guideline when defining a parameter connection:

- Parameter connections must be defined after tasks.

# Examples

**L5X ParameterConnection example**

```
- <Programs>
  - <Program Name="MainProgram" TestEdits="false" MainRoutineName="MainRoutine" Disabled="false"
      UseAsFolder="false">
    - <Tags>
      + <Tag Name="Input_ParameterMain" TagType="Base" DataType="DINT" Radix="Binary" Usage="Input"
          Constant="false" ExternalAccess="Read/Write">
      + <Tag Name="Output_ParameterMain" TagType="Base" DataType="DINT" Radix="Decimal" Usage="Output"
          Constant="false" ExternalAccess="Read Only">
      </Tags>
    + <Routines>
    </Program>
  - <Program Name="SecondProgram" TestEdits="false" Disabled="false" UseAsFolder="false">
    - <Tags>
      + <Tag Name="Input_ParameterFromMain" TagType="Base" DataType="DINT" Radix="Decimal"
          Usage="Input" Constant="false" ExternalAccess="Read/Write">
      + <Tag Name="Input_ParameterFromSub" TagType="Base" DataType="DINT" Radix="Decimal" Usage="Input"
          Constant="false" ExternalAccess="Read/Write">
      </Tags>
      <Routines />
    </Program>
  - <Program Name="SubProgram" TestEdits="false" Disabled="false" UseAsFolder="false">
    - <Tags>
      + <Tag Name="Output_ParameterSub" TagType="Base" DataType="DINT" Radix="Decimal" Usage="Output"
          Constant="false" ExternalAccess="Read Only">
      </Tags>
      <Routines />
    </Program>
  </Programs>
+ <Tasks>
- <ParameterConnections>
    <ParameterConnection EndPoint1="\MainProgram.Output_ParameterMain"
      EndPoint2="\SecondProgram.Input_ParameterFromSub" />
    <ParameterConnection EndPoint1="\SubProgram.Output_ParameterSub"
      EndPoint2="\MainProgram.Input_ParameterMain" />
    <ParameterConnection EndPoint1="\SubProgram.Output_ParameterSub"
      EndPoint2="\SecondProgram.Input_ParameterFromMain" />
  </ParameterConnections>
  <CST MasterID="0" />
```

**L5K PARAMETER_CONNECTION examples**

```
PROGRAM MainProgram (MAIN := "MainRoutine",
                     MODE := 0,
                     DisableFlag := 0,
                     UseAsFolder := 0)
  TAG
    Input_ParameterMain : DINT (RADIX := Binary,
                Usage := Input) := 0;
    Output_ParameterMain : DINT (RADIX := Decimal,
                Usage := Output,
                ExternalAccess := Read Only) := 0;
    Program_tag1 : DINT (RADIX := Decimal,
                Constant := Yes,
                ExternalAccess := Read Only) := 0;
  END_TAG
  ROUTINE MainRoutine
  END_ROUTINE
  CHILD_PROGRAMS
  END_CHILD_PROGRAMS
END_PROGRAM
```

```
PROGRAM SecondProgram (MODE := 0,
                       DisableFlag := 0,
                       UseAsFolder := 0)
  TAG
    Input_ParameterFromMain : DINT (RADIX := Decimal,
                Usage := Input) := 0;
    Input_ParameterFromSub : DINT (RADIX := Decimal,
                Usage := Input) := 0;
  END_TAG
  CHILD_PROGRAMS
  END_CHILD_PROGRAMS
END_PROGRAM
PROGRAM SubProgram (MODE := 0,
                    DisableFlag := 0,
                    UseAsFolder := 0)
  TAG
    Output_ParameterSub : DINT (RADIX := Decimal,
                Usage := Output,
                ExternalAccess := Read Only) := 0;
  END_TAG
  CHILD_PROGRAMS
  END_CHILD_PROGRAMS
END_PROGRAM
TASK MainTask (Type := CONTINUOUS,
               Rate := 10,
               Priority := 10,
               Watchdog := 500,
               DisableUpdateOutputs := No,
               InhibitTask := No)
    MainProgram;
    SubProgram;
    SecondProgram;
END_TASK
PARAMETER_CONNECTION  (EndPoint1 :=
\MainProgram.Output_ParameterMain,
                       EndPoint2 :=
\SecondProgram.Input_ParameterFromSub)
END_PARAMETER_CONNECTION
PARAMETER_CONNECTION  (EndPoint1 :=
\SubProgram.Output_ParameterSub,
                       EndPoint2 :=
\MainProgram.Input_ParameterMain)
END_PARAMETER_CONNECTION
PARAMETER_CONNECTION  (EndPoint1 :=
\SubProgram.Output_ParameterSub,
                       EndPoint2 :=
\SecondProgram.Input_ParameterFromMain)
END_PARAMETER_CONNECTION
CONFIG CST(SystemTimeMasterID := 0) END_CONFIG
```

# Define a trend component

## Introduction

This chapter explains the overall structure of the trend component.

## Trend component

A trend component defines a controller trend. Trend objects are optional. You can have as many as 32 trends per import/export file.

### L5X trend structure

```
<Trends>
      <Trend [Trend_Attributes]>
            <Description>
                  <![CDATA[ text ]]>
            </Description>
            <Template>
                  template_data
            </Template>
            <Pens>
                  pen
            </Pens>
      </Trend>
</Trends>
```

### L5K TREND structure

```
TREND <trend_name> [(Description := "text", Trend_Attributes)]
      Template := [template_data];
      [PEN declaration]
END_TREND
```

## Trend elements

| L5X Item | L5K Item | Description |
|---|---|---|
| N/A | *trend_name* | The name of the trend.<br><br>In L5X, use a Name attribute on the <Trend> element. |
| Description | Description | User information about the trend. |
| Template | *template_data* | The trend template in a byte value list. |
| Pens | PEN | Individual pens within the trend.<br>Each trend can support as many as 8 pens. |

## Trend attributes

| Attribute | Description |
|---|---|
| Name | L5X only. Specify the name of the trend.<br><br>In L5K, the name is an element of the statement. |
| SamplePeriod | Specify how often trending tags are collected in msec (1 msec...30 minutes). |
| NumberOfCaptures | Specifies the maximum number of captures allowed (1...100). |
| CaptureSizeType | Define how the capture size is specified. Type **Samples**, **TimePeriod**, or **NoLimit**. |
| CaptureSize | Specify the number of samples for each capture. The maximum number of samples is 2-hours worth of data samples or 1000 samples, whichever is greater. If the CaptureSizeType is Samples, the range is 1...(2 hours/SamplePeriod) or 1000 samples, whichever is greater. If the CaptureSizeType is TimePeriod, the range is SamplePeriod...2 hours or (SamplePeriod * 1000), whichever is greater. |
| StartTriggerType | Specify the type of the start trigger. Type **NoTrigger** or **EventTrigger**. |
| StartTriggerTag1 | Specify the tag name of the first start trigger. The name must be one of the pen names. |
| StartTriggerOperation1 | Specify the operation that is applied on StartTriggerTag1, and StartTriggerTargetValue1 or StartTriggerTargetTag1.<br>**Enter:** **For:**<br>0 Exact Equal (Tag EQU Target)<br>1 Trigger Level Equal (Tag = Target)<br>2 Not Equal (Tag != Target)<br>3 Less Than (Tag < Target)<br>4 Greater Than (Tag > Target)<br>5 Less Than or Equal To (Tag <= Target)<br>6 Greater Than or Equal To (Tag >= Target)<br>7 Positive Slope (slope of Tag is positive)<br>8 Negative Slope (slope of Tag is negative)<br>9 Bitwise OR ((Tag OR Target) = 0)<br>10 Bitwise OR ((Tag OR Target) != 0)<br>11 Bitwise AND ((Tag AND Target) = 0)<br>12 Bitwise AND ((Tag AND Target) != 0)<br>13 Bitwise XOR ((Tag XOR Target) = 0)<br>14 Bitwise XOR ((Tag XOR Target) != 0) |
| StartTriggerTargetType1 | Specify the type of the first start trigger target. Type **TargetValue** or **TargetTag**. If you type **TargetValue**, **StartTriggerTargetValue1** is expected. Otherwise, **StartTriggerTargetTag1** is expected. |
| StartTriggerTargetValue1 | Specify a target value if the StartTriggerTargetType1 is **TargetValue**. Type a binary, octal, decimal, or hexadecimal integer number or type a floating point number. |
| StartTriggerTargetTag1 | Specify a target tag if the StartTriggerTargetType is **TargetTag**. The tag must be one of the pen names. |
| StartTriggerLogicalOperation | Specify a logical operation (AND or OR) that is performed on StartTriggerxxx1 and StartTriggerxxx2. StartTriggerxxx1 consists of StartTriggerTag1, StartTriggerOperation1, StartTriggerTargetType1, and StartTriggerTargetValue1 or StartTriggerTargetTag1. StartTriggerxxx2 consists of StartTriggerTag2, StartTriggerOperation2, StartTriggerTargetType2, and StartTriggerTargetValue2 or StartTriggerTargetTag2. |
| StartTriggerTag2 | Specify the tag name of the second start trigger. The name must be one of the pen names. |

| Attribute | Description |
|---|---|
| StartTriggerOperation2 | Specify the operation that is applied on StartTriggerTag2, and StartTriggerTargetValue2 or StartTriggerTargetTag2.<br>**Type:** **For:**<br>0 Exact Equal (Tag EQU Target)<br>1 Trigger Level Equal (Tag = Target)<br>2 Not Equal (Tag != Target)<br>3 Less Than (Tag < Target)<br>4 Greater Than (Tag > Target)<br>5 Less Than or Equal To (Tag <= Target)<br>6 Greater Than or Equal To (Tag >= Target)<br>7 Positive Slope (slope of Tag is positive)<br>8 Negative Slope (slope of Tag is negative)<br>9 Bitwise OR ((Tag OR Target) = 0)<br>10 Bitwise OR ((Tag OR Target) != 0)<br>11 Bitwise AND ((Tag AND Target) = 0)<br>12 Bitwise AND ((Tag AND Target) != 0)<br>13 Bitwise XOR ((Tag XOR Target) = 0)<br>14 Bitwise XOR ((Tag XOR Target) != 0) |
| StartTriggerTargetType2 | Specify the type of the second start trigger target. Type **TargetValue** or **TargetTag**. If you type **TargetValue**, **StartTriggerTargetValue2** is expected. Otherwise, **StartTriggerTargetTag2** is expected. |
| StartTriggerTargetValue2 | Specify a target value if the StartTriggerTargetType2 is TargetValue. Type a binary, octal, decimal, or hexadecimal integer number or type a floating point number. |
| StartTriggerTargetTag2 | Specify a target tag if the StartTriggerTargetType is TargetTag. The tag must be one of the pen names. |
| PreSampleType | Define how pre-samples are specified. Type **Samples** or **TimePeriod**. |
| PreSamples | Specify the number of pre-samples (0...1000) if the PreSampleType is Samples. Specify a time period (0...(SamplePeriod * 1000)) that covers pre-samples if the PreSampleType is TimePeriod. |
| StopTriggerType | Specify the type of the stop trigger. Type **NoTrigger** or **Event Trigger**. |
| StopTriggerTag1 | Specify the tag name of the first trigger. The name must be one of the pen names. |
| StopTriggerOperation1 | Specify the operation that is applied on StopTriggerTag1 and StopTriggerTargetValue1 or StopTriggerTargetTag1.<br>**Type:** **For:**<br>0 Exact Equal (Tag EQU Target)<br>1 Trigger Level Equal (Tag = Target)<br>2 Not Equal (Tag != Target)<br>3 Less Than (Tag < Target)<br>4 Greater Than (Tag > Target)<br>5 Less Than or Equal To (Tag <= Target)<br>6 Greater Than or Equal To (Tag >= Target)<br>7 Positive Slope (slope of Tag is positive)<br>8 Negative Slope (slope of Tag is negative)<br>9 Bitwise OR ((Tag OR Target) = 0)<br>10 Bitwise OR ((Tag OR Target) != 0)<br>11 Bitwise AND ((Tag AND Target) = 0)<br>12 Bitwise AND ((Tag AND Target) != 0)<br>13 Bitwise XOR ((Tag XOR Target) = 0)<br>14 Bitwise XOR ((Tag XOR Target) != 0) |
| StopTriggerTargetType1 | Specify the type of the first stop trigger target. Type **TargetValue** or **TargetTag**. If you specify TargetValue, StopTriggerTargetValue1 is expected. Otherwise, StopTriggerTargetTag1 is expected. |
| StopTriggerTargetValue1 | Specify a target value if the StopTriggerTargetType1 is **TargetValue**. Type a binary, octal, decimal, or hexadecimal integer number or type a floating point number. |
| StopTriggerTargetTag1 | Specify a target tag if the StopTriggerTargetType is **TargetTag**. The name must be one of the pen names. |

| Attribute | Description |
|---|---|
| StopTriggerLogicalOperation | Specify a logical operation (AND or OR) that is performed on StopTriggerxxx1 and StopTriggerxxx2. StopTriggerxxx1 consists of StopTriggerTag1, StopTriggerOperation1, StopTriggerTargetType1, and StopTriggerTargetValue1 or StopTriggerTargetTag1. StopTriggerxxx2 consists of StopTriggerTag2, StopTriggerOperation2, StopTriggerTargetType2, and StopTriggerTargetValue2 or StopTriggerTargetTag2. |
| StopTriggerTag2 | Specify the tag name of the second trigger. The name must be one of the pen names. |
| StopTriggerOperation2 | Specify the operation that is applied on StopTriggerTag2 and StopTriggerTargetValue2 or StopTriggerTargetTag2.<br>**Type: For:**<br>0 Exact Equal (Tag EQU Target)<br>1 Trigger Level Equal (Tag = Target)<br>2 Not Equal (Tag != Target)<br>3 Less Than (Tag < Target)<br>4 Greater Than (Tag > Target)<br>5 Less Than or Equal To (Tag <= Target)<br>6 Greater Than or Equal To (Tag >= Target)<br>7 Positive Slope (slope of Tag is positive)<br>8 Negative Slope (slope of Tag is negative)<br>9 Bitwise OR ((Tag OR Target) = 0)<br>10 Bitwise OR ((Tag OR Target) != 0)<br>11 Bitwise AND ((Tag AND Target) = 0)<br>12 Bitwise AND ((Tag AND Target) != 0)<br>13 Bitwise XOR ((Tag XOR Target) = 0)<br>14 Bitwise XOR ((Tag XOR Target) != 0) |
| StopTriggerTargetType2 | Specify the type of the second stop trigger target. Type **TargetValue** or **TargetTag**. If you specify TargetValue, StopTriggerTargetValue2 is expected. Otherwise, StopTriggerTargetTag2 is expected. |
| StopTriggerTargetValue2 | Specify a target value if the StopTriggerTargetType2 is **TargetValue**. Type a binary, octal, decimal, or hexadecimal integer number or type a floating point number. |
| StopTriggerTargetTag2 | Specify a target tag if the StopTriggerTargetType is **TargetTag**. The name must be one of the pen names. |
| PostSampleType | Define how post-samples are specified. Type **Samples** or **TimePeriod**. |
| PostSamples | Specify the number of post-samples (0...1000) if the PostSampleType is Samples. Specify a time period (0...(SamplePeriod * 1000)) that covers post-samples if the PostSampleType is TimePeriod. |
| TrendxVersion | Specify the version of the Trend feature. |

# Pen declaration

A trend object can have as many as eight pen declarations. A pen declaration follows this structure.

## L5X pen structure

```
<Pen [Pen_Attributes]>
      <Description>
            <![CDATA[ text ]]>
      </Description>
</Pen>
```

## L5K PEN structure

```
PEN <pen_name> [(Description := "text", Pen_Attributes)];
END_PEN
```

## Pen elements

| L5X Item | L5K Item | Description |
|---|---|---|
| N/A | *pen_name* | The name of the pen.<br><br>In L5X, use a Name attribute on the <Pen> element. |
| Description | Description | User information about the pen. |

## Pen attributes

| Attribute | Description |
|---|---|
| Name | L5X only. Specify the name of the pen.<br><br>In L5K, the name is an element of the statement. |
| Color | Specify the color of the line in RGB format. Type the hex number for the color (16#0000_0000 – 16#00FF_FFFF). |
| Visible | Specify whether or not the line should be visible. Type **TRUE** or **FALSE**. |
| Width | Specify the width of the line in pixels (1...10). |
| Type | Specify the line type. Type **Analog**, **Digital**, or **Full-Width**. |
| Style | Specify the style of line.<br>**Type:** **For:**<br>0 …….<br>1 … … …. …<br>2 .....<br>3 …. …. …. …<br>4 ….. ….. ….. |
| Marker | Specify the line marker (0...83) |
| Min | Specify the minimum value for the pen. The minimum cannot be greater than or equal to the maximum. |
| Max | Specify the maximum value for the pen. The maximum cannot be less than or equal to the minimum. |
| EngUnits | Specify engineering units. For example, rpm, gallon, fps, and degrees. |

## Trend guidelines

Observe these guidelines when defining a trend:

- A trend can support as many as eight pen declarations.
- Export just the trend of a controller project by right-clicking the trend in the Controller Organizer and choosing **Export**. This saves the trend as a.L5X file (XML format) in the same format as described above for the trend section in the complete project.L5K file.
- To import a trend.L5X file into a controller project, right-click **Trends** in the Control Organizer and select **Import**.

# Examples

**L5X Trend example**

```
- <Trend Name="My_Trend" SamplePeriod="10" NumberOfCaptures="1"
    CaptureSizeType="Samples" CaptureSize="60000" StartTriggerType="No Trigger"
    StopTriggerType="No Trigger" TrendxVersion="5.2">
  - <Description>
      <![CDATA[ trend the controller tags ]]>
    </Description>
    <Template>208 207 17 224 161 177 26 225 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 62 0 3 0
      254 255 9 0 6 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 16 0 0 2 0 0 0 1 0 0 0
      254 255 255 255 0 0 0 0 0 0 0 0 255 255 255 255 255 255 255 255 255 255 255
      255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
      255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
      255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 0 0 0 0 0 0 0 0 0 0 0 0
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 173 60 3 0</Template>
  - <Pens>
      <Pen Name="Local:1:I.Data" Color="16#00ff_0000" Visible="true" Width="1"
        Type="Analog" Style="0" Marker="0" Min="0.0" Max="100.0" />
    </Pens>
  </Trend>
```

**L5K TREND Example**

```
TREND trend1 (SamplePeriod := 10,
              NumberOfCaptures := 1,
              CaptureSizeType := Samples,
              CaptureSize := 60000,
              StartTriggerType := No Trigger,
              StopTriggerType := No Trigger,
              TrendxVersion := 5.2)
Template :=
[208,207,17,224,161,177,26,225,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
62,0,3,0,254,255,9,0,6,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,
0,0,0,0,16,0,0,2,0,0,0,1,0,0,0,254,255,255,255,0,0,0,0,0,0,0,0,0,
255,255,255,255,255,255,....
PEN Local:1:I.CHA_Status (Color := 16#00ff_0000,
                          Visible := 1,
                          Width := 1,
                          Type := Analog,
                          Style := 0,
                          Marker := 0,
                          Min := 0.0,
                          Max := 100.0)
END_PEN
PEN Local:1:I.CHB_Status (Color := 16#0000_ff00,
                          Visible := 1,
                          Width := 1,
                          Type := Analog,
                          Style := 0,
                          Marker := 0,
                          Min := 0.0,
                          Max := 100.0)
END_PEN
END_TREND
```

# Define a watch list component

## Introduction

This chapter explains the overall structure of the watch list component.

## Quick watch list component

The quick watch list component defines a collection of tags that you need to monitor on the fly or for a period of time. A quick watch list is optional. You can have multiple watch lists; each watch list can have multiple tags.

### L5X QuickWatchList structure

```
<QuickWatchLists>
      <QuickWatchList [QuickWatchListAttributes]>
            <WatchTag [WatchTagAttributes] />>
      </QuickWatchList>
</QuickWatchLists>
```

### L5K QUICK_WATCH structure

```
QUICK_WATCH [(Quick_Watch_Attributes)]
      WATCH_TAG [(Watch_Tag_Attributes)] ;
END_QUICK_WATCH
```

### Quick Watch elements

| L5X Item | L5K Item | Description |
|---|---|---|
| QuickWatchLists | N/A | The element that holds quick watch lists. |
| QuickWatchList | QUICK_WATCH | A quick watch list that holds watch tags. |
| WatchTag | WATCH_TAG | An individual watch tag within a watch list. |

### Quick Watch List attributes

| Attribute | Description |
|---|---|
| Name | Specify the name of the quick watch list. |

### Watch tag attributes

| Attribute | Description |
|---|---|
| Specifier | Specify the tag or part of a tag to watch. |
| Scope | Specify the name of program, equipment phase, or Add-On Instruction that contains the watch tag.<br>For L5X, the value is empty if the tag is controller scope.<br>For L5K, the attribute is omitted if the tag is controller scope. |

# Examples

**L5X Quick watch lists example**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RSLogix5000Content ...>
    <Controller ...>
     ...
        <Trends>
            ...
        </Trends>
        <QuickWatchLists>
            <QuickWatchList Name=”NamedQuickWatch_1”>
                    <WatchTag Specifier="MyDint" Scope="" />
                    <WatchTag Specifier="MySint" Scope="My_Program" />
                    <WatchTag Specifier="MyAOI" Scope="My_Program" />
                    <WatchTag Specifier="MyAOI.MyString" Scope="My_Program" />
            </QuickWatchList>
            <QuickWatchList Name=”NamedQuickWatch_2”>
                    <WatchTag Specifier="MyDint" Scope="" />
                    <WatchTag Specifier="MySint" Scope="My_Program" />
                    <WatchTag Specifier="MyAOI" Scope="My_Program" />
                    <WatchTag Specifier="MyAOI.MyString" Scope="My_Program" />
             </QuickWatchList>
        </QuickWatchLists>
    </Controller>
</RSLogix5000Content>
```

**L5K QUICK_WATCH example**

```
QUICK_WATCH (Name := My_Quick_Watch_2)
       WATCH_TAG (Specifier := MyDint);
       WATCH_TAG (Specifier := MySint, Scope := My_Program);
       WATCH_TAG (Specifier := MyAOI, Scope := My_Program);
       WATCH_TAG (Specifier := MyAOI.MyString, Scope :=
MyProgram);
END_QUICK_WATCH

QUICK_WATCH (Name := My_Quick_Watch_1)
       WATCH_TAG (Specifier := MyDint);
       WATCH_TAG (Specifier := MySint, Scope := My_Program);
       WATCH_TAG (Specifier := MyAOI, Scope := My_Program);
       WATCH_TAG (Specifier := MyAOI.MyString, Scope :=
MyProgram);
END_QUICK_WATCH
```

# Define controller configuration objects

## Introduction

This chapter explains how to enter project and configuration information in a complete import/export file.

## Controller objects

The config component defines controller objects.

### L5X config structure

```
<CommPorts>
      <SerialPort [SerialPort_Attributes]>
            <ASCII [ASCII_Attributes]/>
            <DF1 [DF1_Attributes]/>
      </SerialPort>
</CommPorts>
<CST [CST_Attributes]/>
<WallClockTime [WallClockTime_Attributes]/>
```

### L5K CONFIG structure

```
CONFIG <object_name> [(Object_Attributes)]
END_CONFIG
```

### Object elements

| L5X Item | L5K Item | Description |
|---|---|---|
| N/A | *object_name* | The name of the controller config object.<br><br>In L5X, each controller config object is specified with an element specific to that object. |

Controller objects are optional. There can be only one of each controller object in an import/export file. Controller objects appear at the end of the import/export file.

## Config attributes

The attributes depend on the type on the object. Some objects do not have any attributes.

| Object | Attribute | Description |
|---|---|---|
| ASCII | XONXOFFEnable | Specify whether to regulate the flow of incoming data. Type **0** to disable XON/XOFF; type **1** to enable XON/XOFF. |
| | DeleteMode | Specify the delete mode. Type **0** for Ignore; type **1** for CRT; or type **2** for Printer. |
| | EchoMode | Specify whether to echo data back to the device from which it was sent. Type **0** to disable; type **1** to enable. |
| | TerminationChars | Specify the characters that designate the end of a line. |

| Object | Attribute | Description |
|---|---|---|
| | AppendChars | Specify the characters to append to the end of a line. |
| | BufferSize | Specify the maximum size of the data array (1...65535 bytes) to send and receive. |
| CST | SystemTimeMasterIDn (L5K)<br>MasterID (L5X) | Specify whether the controller is the coordinated system time master. Type **16#0000** if the controller is not the CST master; type **16#0001** if the controller is the CST master. |
| DF1 | DuplicateDetection | Specify whether to enable duplicate message detection, which ignores duplicate messages. Type **0** to disable; type **1** to enable. |
| | ErrorDetection | Specify the error detection method. Type **BCC Error** or **CRC Error**. |
| | EnbeddedResponseEnable | Specify the response method. Type **0** to autodetect; type **1** to enable. |
| | DF1Mode | Specify the DF1 mode. Type **Pt to Pt**, **Master**, or **Slave**. |
| | ACKTimeout | Specify the time to wait for an acknowledgment to a message transmission. Type an increment of 20 ms (0...32767). |
| | NAKReceiveLimit | Specify the number of NAKS (0...127) the controller can receive in response to a message before stopping transmission. |
| | ENQTransmit | Specify the number of inquiries (0...127) the controller sends after an ACK timeout. |
| | TransmitRetries | Specify the number of attempted retries (0...127) without getting an acknowledgment before the message is deemed undeliverable. |
| | StationAddress | Specify the current station link address (0...254). |
| | ReplyMessageWait | Specify the time the master waits after receiving an acknowledgment to a master-initiated message before polling the slave for a response. Type an increment of 20 ms (0...65535). |
| | PollingMode | Specify the polling mode. Type one of these:<br>• 1 for Message Based (slave can initiate messages)<br>• 2 for Message Based (slave cannot initiate messages)<br>• 3 for Standard (multiple message transfer for node scan)<br>• 4 for Standard (single message transfer per node scan) |
| | MasterMessageTransmit | Specify when the master transmits. Type **0** to transmit between station polls; type **1** to transmit in poll sequence. |
| | NormalPollNodeFile | Specify the tag name of the structure that contains the normal poll node list, or type **<NA>**. The tag must specify Class = Standard. |
| | NormalPollGroupSize | Specify the total number (0...255) of active stations polled from the poll list. |
| | PriorityPollNodeFile | Specify the tag name of the structure that contains the priority poll node list, or type **<NA>**. The tag must specify Class = Standard. |
| DF1 | ActiveStationFile | Specify the tag name of the structure that contains the status (active or non-active) of each node, or type **<NA>**. The tag must specify Class = Standard. |
| | SlavePollTimeout | Specify the amount of time the master waits for an acknowledgment to a message sent to a slave. Type an increment of 20 ms (0...65535). |
| | EOTSuppression | Specify whether to enable EOT suppression. Type **0** to disable; type **1** to enable. |
| | MaxStationAddress | Specify the maximum station address (0...31). |
| | TokenHoldFactor | Specify the token hold factor (1...4). |
| | EnableStoreFwd | For DF1 radio modem, specify whether to enable the store and forward feature. Type **0** to disable; type **1** to enable. |
| | StoreFwdFile | Specify the INT tag that holds the store and forward table. |
| SerialPort | Channel (L5X only) | Specify the serial port. |

| Object | Attribute | Description |
|---|---|---|
| | BaudRate | Specify the communication rate for the serial port. Type **110, 300 600, 1200, 2400, 4800, 9600, 19200, or 38400**. |
| | Parity | Specify the parity setting for the serial port. Parity provides additional message-packet error detection. Type **None Parity**, **Even Parity**, or **Odd Parity**. |
| | DataBits | Specify the number of bits per message packet. Type **7 Data Bits** or **8 Data Bits**. |
| | StopBits | Specify the number of stop bits to the device with which the controller is communicating. Type **1 Stop Bit** or **2 Stop Bit**. |
| | ComDriverId | Specify the type of serial driver. Type **DF1**. |
| | PendingComDriverId | L5K only. Specify type of serial driver. Type **DF1**. |
| | RTSOffDelay | Specify a time delay to make sure the modem successfully transmits the entire message. Type an increment of 20 ms (0...32767). Normally leave at zero. |
| | RTSSendDelay | Specify a time delay to let the modem prepare to transmit a message. Type an increment of 20 ms (0...32767). |
| | ControlLine | Specify the mode in which the serial driver operates. Type **No Handshake**, **Full Duplex**, **Half Duplex without Continuous Carrier**, or **Half Duplex with Continuous Carrier**. |
| | PendingControlLine | L5K only. Specify the mode in which the serial driver operates. Type **No Handshake**, **Full Duplex**, **Half Duplex without Continuous Carrie**r, or **Half Duplex with Continuous Carrier**. |
| | RemoteModeChangeFlag | Specify whether there is a remote change. Type **0** or **1**. |
| | PendingRemoteModeChangeFlag | L5K only. Specify whether there is a remote change. Type **0** or **1**. |
| | ModeChangeAttentionChar | Specify the mode change attention character. |
| | PendingModeChangeAttentionChar | L5K only. Specify the mode change attention character. |
| SerialPort | SystemModeCharacter | Specify the system mode character. |
| | PendingSystemModeCharacter | L5K only. Specify the system mode character. |
| | UserModeCharacter | Specify the user mode character. |
| | PendingSystemModeCharacter | L5K only. Specify the user mode character. |
| Serial Port (cont) | DCDWaitDelay | For DF1 radio modem, specify the delay in seconds (0...255). Specify this value if ControlLine is Half-Duplex and ContinuousCarrier is disabled. |
| WallClockTime | LocalTimeAdjustment | Specify any local time adjustment. |
| | TimeZone | Specify the time zone. |
| Internet Protocol | ConfigType | Specify the IP configuration type.<br><br>Values include **Manual**, **BOOTP**, or **DHCP**. |
| | IPAddress | Specify the IP Address. |
| | SubnetMask | Specify the Subnet Mask. |
| | Gateway | Specify the Gateway. |
| | PrimaryDNS | Specify the Primary DNS. |
| | SecondaryDNS | Specify the Secondary DNS. |
| | DomainName | Specify the Domain Name. |
| | HostName | Specify the Host Name. |
| Ethernet Port | Port | This is the Ethernet Port number being configured. (L5X only) |
| | PortEnabled | Specifies if the port is Enabled or Disabled. |

| Object | Attribute | Description |
|---|---|---|
| | AutoNegotiateEnabled | Specifies if Auto Negotiate is Enabled or Disabled. |
| | InterfaceSpeed | Specifies the Interface Speed in Mbps.<br><br>Valid values are 10 or 100. |
| | DuplexMode | Specifies the Duplex Mode.<br><br>Valid values are Half or Full. |
| Ethernet Network | SupervisorModeEnabled | Specify if the ring supervisor mode of the controller is Enabled. |
| | SupervisorPrecedence | Specifies the Supervisor Precedence.<br><br>A numerically higher value indicates higher precedence.<br><br>Values are in the range of 0...255. |
| | BeaconInterval | Specifies the Beacon Interval for the ring.<br><br>Values are in the range of 100...100000 usec. |
| | BeaconTimeout | Specifies the Beacon Timeout for the ring.<br><br>Values are in the range of 200...500000 usec. |
| | VLANID | Specifies the Ring Protocol VLAN ID of the ring.<br><br>Values are in the range of 0... 4094 |

# Examples

**L5X Config example**

```
- <CommPorts>
  - <SerialPort Channel="0" BaudRate="19200" Parity="No Parity" DataBits="8 Bits of Data"
      StopBits="1 Stop Bit" ComDriverId="DF1" RTSOffDelay="0" RTSSendDelay="0"
      ControlLine="No Handshake" RemoteModeChangeFlag="false"
      ModeChangeAttentionChar="27" SystemModeCharacter="83" UserModeCharacter="85"
      DCDWaitDelay="0">
      <ASCII XONXOFFEnable="false" DeleteMode="0" EchoMode="0"
        TerminationChars="65293" AppendChars="2573" BufferSize="82" />
      <DF1 DuplicateDetection="true" ErrorDetection="BCC Error"
        EmbeddedResponseEnable="Autodetect" DF1Mode="Pt to Pt" ACKTimeout="50"
        NAKReceiveLimit="3" ENQTransmitLimit="3" TransmitRetries="3" StationAddress="0"
        ReplyMessageWait="5" PollingMode="Message Based (slave can initiate messages)"
        MasterMessageTransmit="Between station polls" NormalPollNodeFile="<NA>"
        NormalPollGroupSize="0" PriorityPollNodeFile="<NA>" ActiveStationFile="<NA>"
        SlavePollTimeout="3000" EOTSuppression="0" MaxStationAddress="31"
        TokenHoldFactor="1" EnableStoreFwd="false" StoreFwdFile="<NA>" />
    </SerialPort>
  </CommPorts>
  <CST MasterID="0" />
  <WallClockTime LocalTimeAdjustment="0" TimeZone="0" />
```

#### L5K CONFIG Examples

This example shows a DF1 controller object.

```
CONFIG DF1
       DuplicateDetection := -1,
       ErrorDetection := BCC Error,
       EmbeddedResponseEnable := -1,
       DF1Mode := Pt to Pt,
       ACKTimeout := 50,
       NAKReceiveValue := 3,
       DF1ENQs := 3,
       DF1Retries := 3,
       StationAddress := 0,
       ReplyMessageWait := 50,
       PollingMode := 0,
       MasterMessageTransmit := 0,
       NormalPollNodeFile := NA,
       NormalPollGroupSize := 0,
       PriorityPollNodeFile := NA,
       ActiveStationFile := NA)
END_CONFIG
```

This example shows a SerialPort controller object.

```
CONFIG SerialPort
       (BaudRate := 19200,
       Parity := No Parity,
       DataBits := 8 Bits of Data,
       StopBits := 1 Stop Bit,
       ComDriverId := DF1,
       RTSOffDelay := 0,
       RTSSendDelay := 0,
       ControlLine := No Handshake,
       RemoteModeChangeFlag := 0,
       ModeChangeAttentionChar := 27,
       SystemModeCharacter := 83,
       UserModeCharacter := 85)
END_CONFIG
```

# Define custom properties

## Introduction

This chapter explains how to define custom properties in a controller project.

Custom properties can be used to store any additional information that should persist with the project. This information is imported with the project, downloaded to the controller, uploaded from the controller, and exported with the project. Custom properties can be added to most XML elements in the L5X format. For example, you can add it to the controller, data types, data type members, modules, add-on instructions, tags, programs, routines, rungs, sheets, various instructions, various collections, ST lines, trends, and so on. However, you cannot add custom properties to custom properties. Custom properties are compatible only with the L5X format.

Custom property data is well-formed XML limited to chunks of 65000 bytes. To associate more thank 65K of data on a single parent object, break it into multiple sections using the extension attribute. This optional attribute is a free-form string, so you can organize custom properties information in any way you need.

## Custom properties data

The following sections describe custom properties data.

### Custom properties structure

Custom Properties should be the first child xml element under the parent element.

```
<ParentXmlObject … >
      <CustomProperties>
            <Provider ID="YourCompanyName" Ext="Part 1" >
                  <YourData YourAttrib="Your Value" />
                  <YourOtherData Attrib2="2" >
                        <YourChildElement Attrib3="3" />
                  </YourOtherData>
            </Provider>
            <Provider ID="YourCompanyName" Ext="Part 2" >
                  <![CDATA[ Your custom property data wrapped
in CData ]]>
            </Provider>
            <Provider ID="ToolMakerX" >
                  <ToolMakerData >Custom property data from a
```

```
tool you are using</ToolMakerData>
            </Provider>
      </CustomProperties>
      … (the rest of the ParentObject's child elements) …
</ParentXmlObject >
```

## Custom properties elements

| L5X item | Description |
|---|---|
| CustomProperties | The collection of Provider elements. |
| Provider | Unique element that wraps the user's custom property XML. The XML between this beginning and end element must be well-formed. |

## Custom properties attributes

| Attribute | Description |
|---|---|
| ID | The provider attribute which identifies the owner of this custom property data. This is a free-form string. It could identify your company, or it might come from a third party (OEM) component or tool that you are using. |
| Ext | (Optional) This extension attribute flags custom property data when you need to break it into multiple Provider sections. This is a free-form string. An attribute with an empty string is the same as not having the attribute. |

## Example

The following example demonstrates adding custom properties for a tag:

```
<Tag Name="MyTag" TagType="Base" DataType="DINT"
Radix="Decimal" Constant="false" ExternalAccess="Read/Write">
      <CustomProperties>
            <Provider ID="CompanyX" >
                  <CompanXTextData > Your Xml data as a Text
Node </CompanXTextData>
            </Provider>
      </CustomProperties>
      <Data Format="L5K">
            <![CDATA[42]]>
      </Data>
      <Data Format="Decorated">
            <DataValue DataType="DINT" Radix="Decimal"
Value="42"/>
      </Data>
</Tag>
```

# Structure Tags and Comments in an Import/Export File

## Introduction

This chapter explains how to structure the import/export file by using commas (in a CSV text file) or tabs (in a TXT Unicode text file) to separate values in the file.

## Place information in a.CSV or.TXT file

The structured import/export file contains these components for comments and other information.

| Item | Identifies |
|---|---|
| Remark | Comment within the file. |
| TAG | Tag. |
| RCOMMENT | Rung comment. |
| TEXTBOX | Text box comment. |

## Internal file comments

Enter comments to document import files. The import process ignores these comments. Place comments anywhere in an import/export file, except in names and descriptions. Enter comments by starting the line (record) with REMARK and a comma.

## Specify a tag record

Each tag record defines a tag within a controller project. A tag record includes this information.

| Item | Identifies |
|---|---|
| Type | The type of tag.<br>TAG tag<br>ALIAS alias tag<br>COMMENT tag operand component |
| Scope | The part of the project that owns the tag.<br>• If no scope is specified, the scope is controller.<br>• If a scope is specified, it identifies the program or equipment phase. |
| Name | Name of the tag |
| Description | Description of the tag (optional) |
| Datatype | Datatype of the tag - use any valid datatype name |

| Item | Identifies |
|---|---|
| Specifier | Optional<br>• An alias, specifies base tag.<br>• A tag comment, specifies the tag name and member or bit. |
| Attributes | The attributes of the tag, as exported in the L5K format.<br>Define how the tag can be used and how it appears.<br>Attributes do not include tag values. |

## TAG type record

Each TAG record defines a tag within a controller project.

### TAG Structure with Commas

```
TAG,"Scope","Name","Description","Datatype","Specifier","At
tributes"
```

### TAG Structure with Tabs

```
TAG,"Scope"  "Name"  "Description"  "Datatype"  "Specifier"
"Attributes"
```

Specify tag dimensions on the Datatype.

| To specify: | Type: |
|---|---|
| 1 dimension | [a] |
| 2 dimensions | [a,b] |
| 3 dimensions | [a,b,c] |

This example shows TAG records in a CSV format.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 7 | TYPE | SCOPE | NAME | DESCRIPTION | DATATYPE | SPECIFIER | ATTRIBUTES | | |
| 8 | TAG | | Local:1:C | | AB:1756_DI:C:0 | | | | |
| 9 | TAG | | Local:1:I | | AB:1756_DI:I:0 | | | | |
| 10 | TAG | | Local:4:C | | AB:1756_DO:C:0 | | | | |
| 11 | TAG | | Local:4:I | | AB:1756_DO:I:0 | | | | |
| 12 | TAG | | Local:4:O | | AB:1756_DO:O:0 | | | | |
| 13 | ALIAS | | input_1 | | | Local:1:I.Data.20 | (RADIX := Decimal) | | |
| 14 | TAG | | MC_Reset | | MC_Config | | | | |
| 15 | ALIAS | | output_light | | | Local:4:O.Data.12 | (RADIX := Decimal) | | |
| 16 | TAG | | Reset4 | | MESSAGE | | (MessageType := CIP Generic, RequestedLeng | | |
| 17 | TAG | | Write_MC_Cfg9 | | MESSAGE | | (MessageType := CIP Generic, RequestedLeng | | |
| 18 | TYPE | SCOPE | NAME | DESCRIPTION | DATATYPE | SPECIFIER | ATTRIBUTES | | |
| 19 | TAG | MainProgram | HeartBeat | | TIMER[2] | | | | |
| 20 | TAG | MainProgram | HeartBeatWord | | DINT | | (RADIX := Decimal) | | |
| 21 | TYPE | SCOPE | NAME | DESCRIPTION | DATATYPE | SPECIFIER | ATTRIBUTES | | |
| 22 | TAG | Multicast_Configure | Configure_Multicast | | BOOL | | (RADIX := Decimal) | | |
| 23 | TAG | Multicast_Configure | DoNothing | | BOOL | | (RADIX := Decimal) | | |
| 24 | TAG | Multicast_Configure | MC_Cfg_TTL_over_1 | | BOOL | | (RADIX := Decimal) | | |
| 25 | TAG | Multicast_Configure | Read_Mcast | | DINT | | (RADIX := Decimal) | | |

## ALIAS type record

Each ALIAS record defines an alias within a controller project.

### ALIAS Structure with Commas

```
ALIAS,"Scope","Name","Description","Datatype","Specifier","
Attributes"
```

### ALIAS Structure with Tabs

```
ALIAS  "Scope"  "Name"  "Description"  "Datatype"
"Specifier"  "Attributes"
```

This example shows ALIAS records in a CSV format.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | TYPE | SCOPE | NAME | DESCRIPTION | DATATYPE | SPECIFIER | ATTRIBUTES |
| 8 | ALIAS | | input_1 | | | Local:1:I.Data.20 | (RADIX := Decimal) |
| 9 | ALIAS | | output_light | | | Local:4:O.Data.12 | (RADIX := Decimal) |

## COMMENT type record

Each COMMENT record defines a comment about a component of a tag, such as a bit member, structure member, or an array element.

### COMMENT Structure with Commas

```
COMMENT,"Scope","Name","Description","Datatype","Specifier"
,"Attributes"
```

### COMMENT Structure with Tabs

```
COMMENT  "Scope"  "Name"  "Description","Datatype"
"Specifier"  "Attributes"
```

This example shows COMMENT records in a CSV format.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 9 | TYPE | SCOPE | NAME | DESCRIPTION | DATATYPE | SPECIFIER | ATTRIBUTES |
| 10 | TAG | | array_1 | | DINT[10] | | (RADIX := Decimal) |
| 11 | COMMENT | | array_1 | first element in array_1 | | array_1[0] | |
| 12 | TAG | | timer_1 | | TIMER | | |
| 13 | COMMENT | | timer_1 | timer_1 enable | | timer_1.EN | |

## Specify a comment record

Each comment record defines a rung comment or text box in the controller project. This is different than the comment type that defines a comment about a tag component. A comment record includes this information.

| Item | Identifies |
|---|---|
| Type | The type of comment.<br>RCOMMENT ladder rung comment<br>TEXTBOX function block or sequential function chart comment |
| Scope | The part of the project that owns the comment.<br>A program or equipment phase must be specified. |
| Routine | Name of the routine. |
| Comment | Text of the comment. |

| Item | Identifies |
|---|---|
| Owning Element | For RCOMMENT entries, neutral text for the last instruction on the rung that owns the comment.<br>If there is no element on the rung, the Owning Element is a semi-colon (;).<br>By default, the Owning Element is used to match the comment to a rung on import.<br>For a TEXTBOX entry of an attached text box, neutral text identifies the element attached to the the text box. The Owning Element contains the backing tag name and the full specifier of the element, including the absolute location of the element.<br>Owning Element.<br>For a TEXTBOX entry of a free-floating text box, this entry is blank. |
| Location | For RCOMMENT entries, the rung number of comment. The rung number in the Location column is used to match the comment to a rung if the Owning Element is blank for that comment or if you override the import default by selecting Match all RLL rung comments by rung number only.<br>For TEXTBOX entries, the absolute location of free-floating text boxes or the relative location from the owning element of attached text boxes. For absolute locations, the location contains both the sheet number and the X and Y coordinates of the text box. For relative locations, the location contains only the X and Y coordinates. |

An RCOMMENT record follows this format.

## RCOMMENT Structure with Commas

```
RCOMMENT,"Scope","Routine","Comment","Owning Element","Location"
```

## RCOMMENT Structure with Tabs

```
RCOMMENT "Scope" "Routine" "Comment" "Owning Element" "Location"
```

A TESTBOX record follows this format.

## TEXTBOX Structure with Commas

```
TEXBOX,"Scope","Routine","Comment","Owning Element","Location"
```

## TEXTBOX Structure with Tabs

```
TEXTBOX "Scope" "Routine" "Comment" "Owning Element" "Location"
```

This example shows comment records in a CSV format.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 118 | TYPE | SCOPE | ROUTINE | COMMENT | OWNING_ELEMENT | LOCATION | |
| 119 | TEXTBOX | Motor_Starter_Progr | Motor_Starter_FBD | Gives simple conveyor control such as start, stop, jog, and $ | | 60:20:Sheet1 | |
| 120 | RCOMMENT | Motor_Starter_Progr | Motor_Starter_LD | If Job_PB = on, then jog the conve | OTE(Motor_Starter_LD.J | | 0 |
| 121 | TEXTBOX | Motor_Starter_Progr | Motor_Starter_SFC | SFC Execution Configuration$N$NLast Scan of Active Steps | | 360:40:00 | |
| 122 | RCOMMENT | Motor_Starter_Progr | Nested_Motor_Sta | Conveyor Control$NThese rungs ar | NOP() | | 0 |
| 123 | TYPE | SCOPE | ROUTINE | COMMENT | OWNING_ELEMENT | LOCATION | |
| 124 | RCOMMENT | Sort_Program | Fill_Init_Numbers | Fills the Init_Numbers array with 1 | OTU(Fill_Numbers) | | 0 |
| 125 | TYPE | SCOPE | ROUTINE | COMMENT | OWNING_ELEMENT | LOCATION | |
| 126 | RCOMMENT | Motor_Starter:AOI | Logic | If the Stop button is closed, the mo | OTE(RunCommand) | | 0 |
| 127 | RCOMMENT | Motor_Starter:AOI | Logic | The motor starts if the run commar | OTE(Out) | | 1 |
| 128 | RCOMMENT | Motor_Starter:AOI | Logic | If FaultTime is greater than 0, turn | OTE(CheckAuxContact) | | 2 |
| 129 | RCOMMENT | Motor_Starter:AOI | Logic | If CheckAuxContact is on, the rung | OTL(Fault) | | 3 |
| 130 | RCOMMENT | Motor_Starter:AOI | Logic | To clear the fault of the motor, turn | OTU(Fault) | | 4 |

## Specify an alarm message record

An alarm tag can have several alarm message strings for different alarm conditions in different languages. An alarm message record includes this information.

| Item | Identifies |
|---|---|
| Type | The alarm message and its associated language as: ALMMSG:*language*<br><br>Languages: **EN-US** (United States English), **DE** (Germany German), **ES** (Spain Spanish), **FR** (France French), **IT** (Italian), **PT** (Brazil Portuguese), **JA** (Japanese), **KO** (Korean), **ZH** (Chinese) |
| Scope | The part of the project that owns the comment.<br>A program or equipment phase must be specified. |
| Name | Name of the associated alarm tag. |
| Description | Text of the alarm message. |
| Datatype | The type of alarm. Specify **ALARM_DIGITAL** or **ALARM_ANALOG**. |
| Specifier | Specify the type of alarm.<br>**Specify:** **For:**<br>AM Digital alarm<br>HH High-high analog alarm<br>H High analog alarm<br>L Low analog alarm<br>LL Low-low analog alarm<br>POS Rate-of-change positive analog alarm<br>NEG Rate-of change negative analog alarm |

An ALMMSG record follows this format.

## ALMMSG Structure with Commas

```
      ALMMS:language,"Scope","Name","Description","Datatype","
Specifier"
```

## ALMMSG Structure with Tabs

```
             ALMMSG:language  "Scope"  "Name"  "Description"
"Datatype"  "Specifier"
```

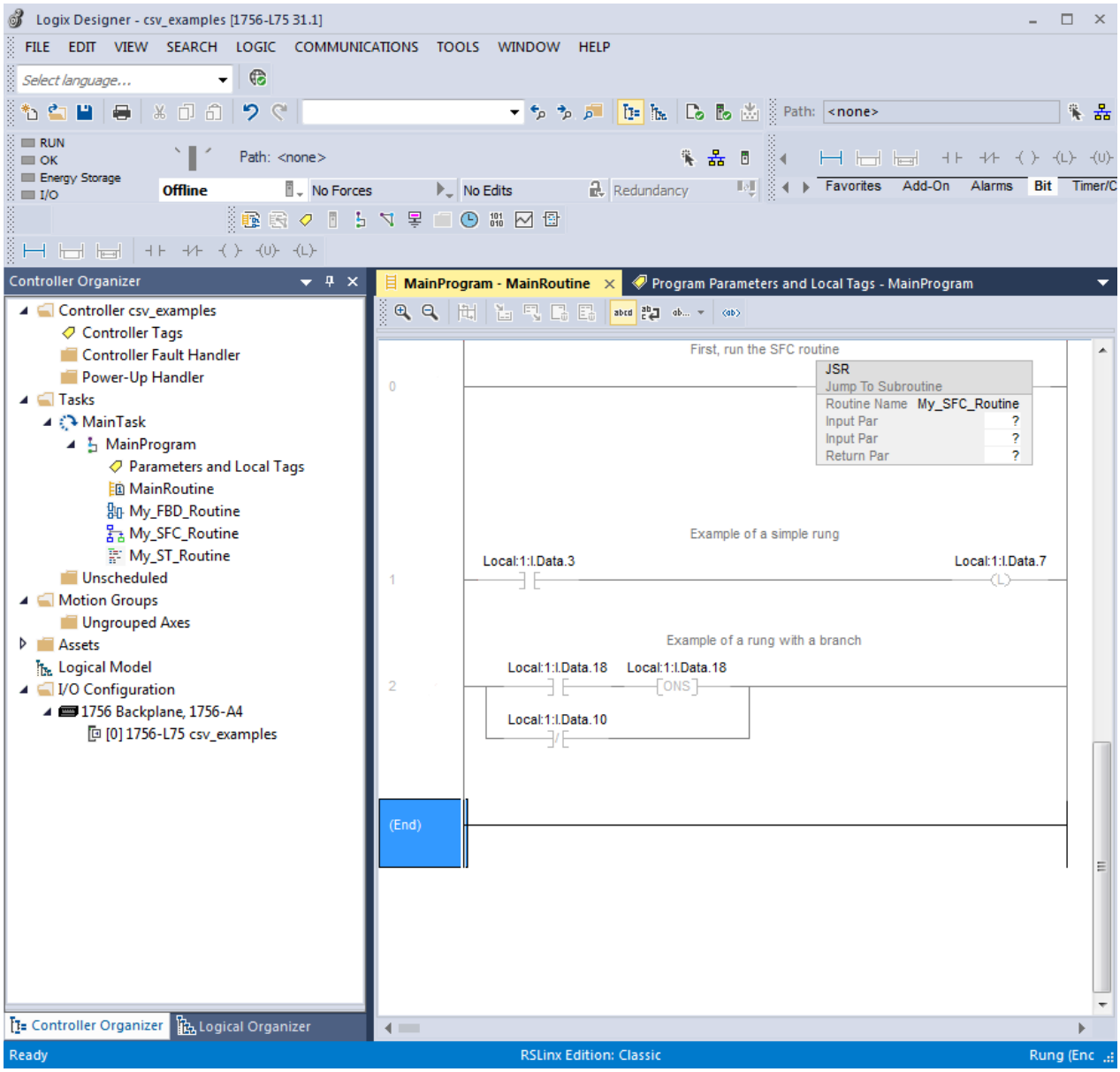
This example shows alarm message records in a CSV format.

| | | | | | | |
|---|---|---|---|---|---|---|
| 39 | TAG | Motor_Starter_Program | ALMA_01 | | ALARM_ANALOG | |
| 40 | ALMMSG:en-us | Motor_Starter_Program | ALMA_01 | high-message | | HH |
| 41 | ALMMSG:fr | Motor_Starter_Program | ALMA_01 | message in french | | POS |
| 42 | ALMMSG:de | Motor_Starter_Program | ALMA_01 | message in german | | NEG |
| 43 | TAG | Motor_Starter_Program | ALMD_01 | | ALARM_DIGITAL | |
| 44 | ALMMSG:en-us | Motor_Starter_Program | ALMD_01 | digital message | | AM |

# Examples

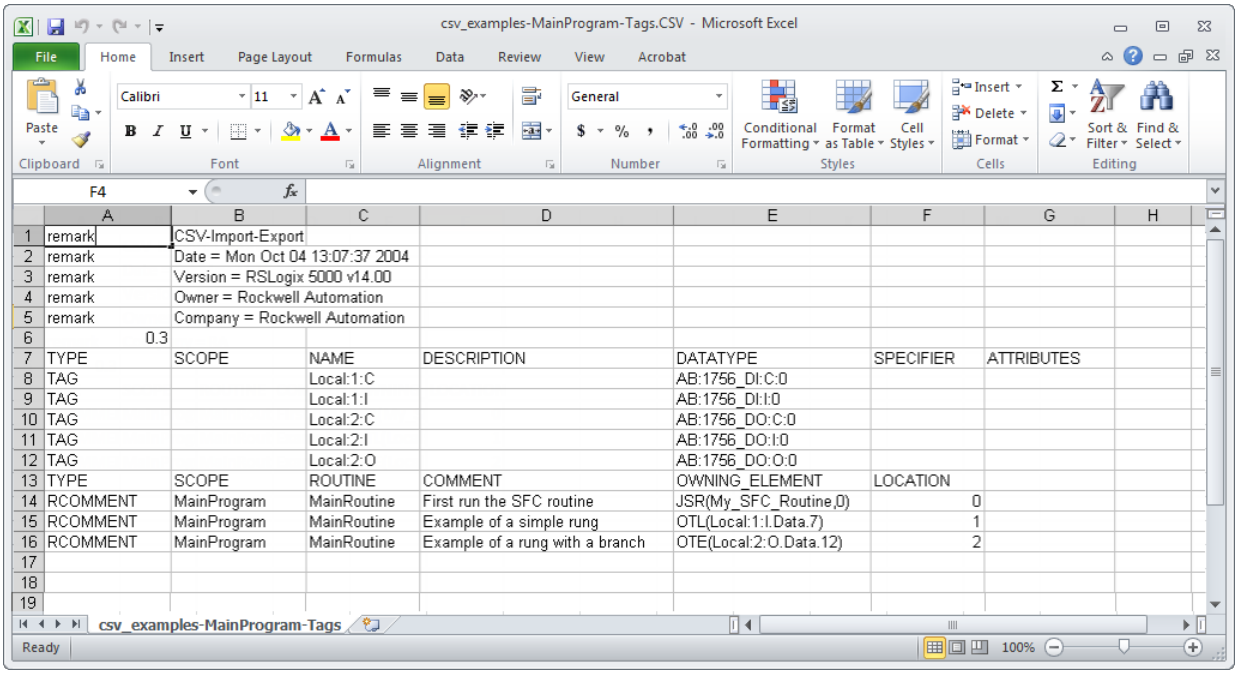
### Example.CSV file

These examples use this ladder file.



### Export all tags and comments

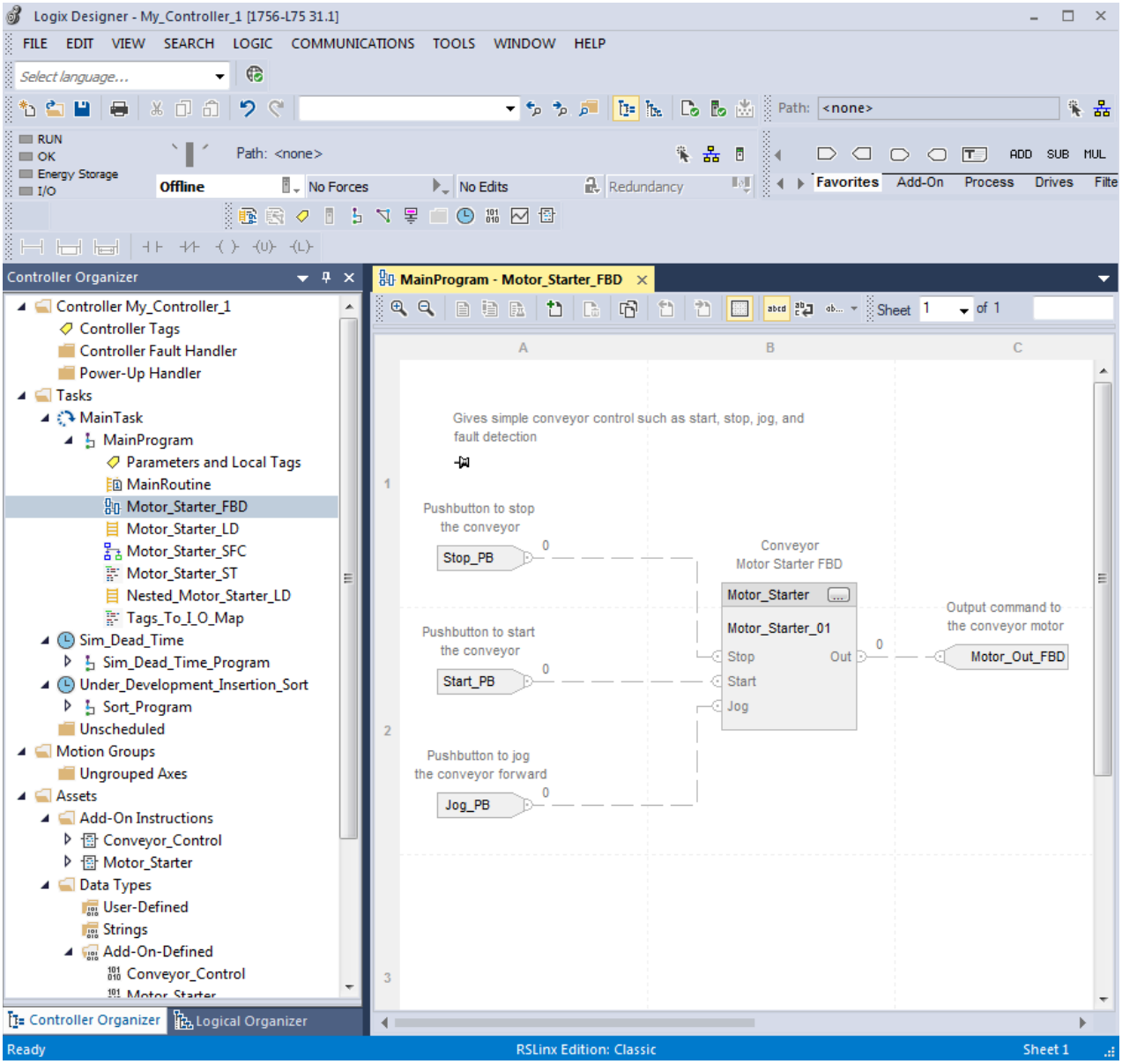
An export of all tags and comments results in this.CSV file.




| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | remark | CSV-Import-Export | | | | | | |
| 2 | remark | Date = Mon Oct 04 13:07:37 2004 | | | | | | |
| 3 | remark | Version = RSLogix 5000 v14.00 | | | | | | |
| 4 | remark | Owner = Rockwell Automation | | | | | | |
| 5 | remark | Company = Rockwell Automation | | | | | | |
| 6 | 0.3 | | | | | | | |
| 7 | TYPE | SCOPE | NAME | DESCRIPTION | DATATYPE | SPECIFIER | ATTRIBUTES | |
| 8 | TAG | | Local:1:C | | AB:1756_DI:C:0 | | | |
| 9 | TAG | | Local:1:I | | AB:1756_DI:I:0 | | | |
| 10 | TAG | | Local:2:C | | AB:1756_DO:C:0 | | | |
| 11 | TAG | | Local:2:I | | AB:1756_DO:I:0 | | | |
| 12 | TAG | | Local:2:O | | AB:1756_DO:O:0 | | | |
| 13 | TYPE | SCOPE | ROUTINE | COMMENT | OWNING_ELEMENT | LOCATION | | |
| 14 | RCOMMENT | MainProgram | MainRoutine | First run the SFC routine | JSR(My_SFC_Routine,0) | 0 | | |
| 15 | RCOMMENT | MainProgram | MainRoutine | Example of a simple rung | OTL(Local:1:I.Data.7) | 1 | | |
| 16 | RCOMMENT | MainProgram | MainRoutine | Example of a rung with a branch | OTE(Local:2:O.Data.12) | 2 | | |
| 17 | | | | | | | | |
| 18 | | | | | | | | |
| 19 | | | | | | | | |

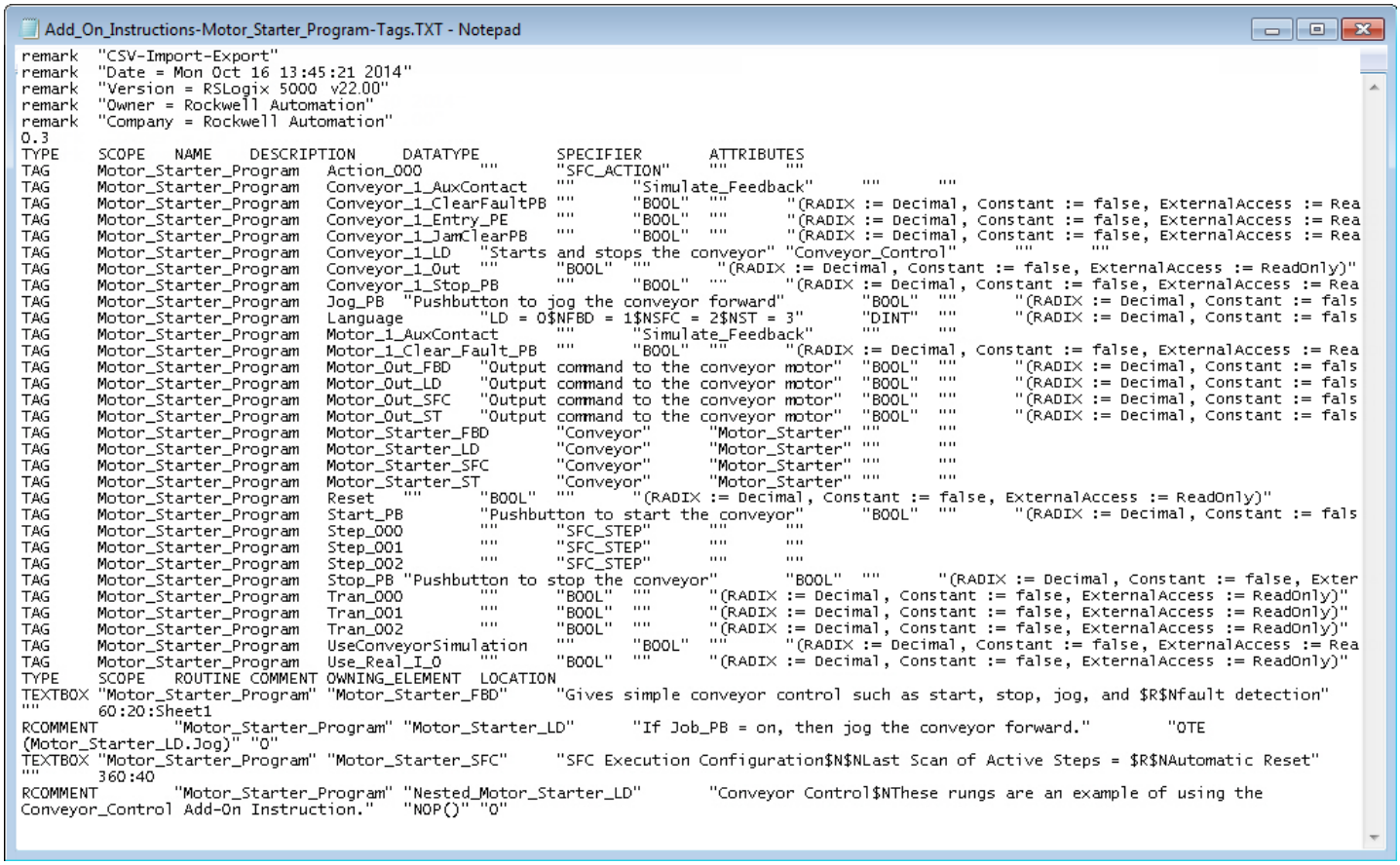
**Example.TXT file**

These examples use the Motor_Starter_Program program file and exports the program parameters and local tags.



**Export program tags and comments**

An export of the Motor_Starter_Prorgam program tags and comments results in this.TXT file.



Add_On_Instructions-Motor_Starter_Program-Tags.TXT - Notepad

```
remark  "CSV-Import-Export"
remark  "Date = Mon Oct 16 13:45:21 2014"
remark  "Version = RSLogix 5000 v22.00"
remark  "Owner = Rockwell Automation"
remark  "Company = Rockwell Automation"
0.3
TYPE    SCOPE   NAME    DESCRIPTION     DATATYPE        SPECIFIER       ATTRIBUTES
TAG     Motor_Starter_Program   Action_000      ""      "SFC_ACTION"    ""      ""
TAG     Motor_Starter_Program   Conveyor_1_AuxContact   ""      "Simulate_Feedback"     ""      ""
TAG     Motor_Starter_Program   Conveyor_1_ClearFaultPB ""      "BOOL"  ""      "(RADIX := Decimal, Constant := false, ExternalAccess := Rea
TAG     Motor_Starter_Program   Conveyor_1_Entry_PE     ""      "BOOL"  ""      "(RADIX := Decimal, Constant := false, ExternalAccess := Rea
TAG     Motor_Starter_Program   Conveyor_1_JamClearPB   ""      "BOOL"  ""      "(RADIX := Decimal, Constant := false, ExternalAccess := Rea
TAG     Motor_Starter_Program   Conveyor_1_LD   "Starts and stops the conveyor" "Conveyor_Control"      ""      ""
TAG     Motor_Starter_Program   Conveyor_1_Out  ""      "BOOL"  ""      "(RADIX := Decimal, Constant := false, ExternalAccess := ReadOnly)"
TAG     Motor_Starter_Program   Conveyor_1_Stop_PB      ""      "BOOL"  ""      "(RADIX := Decimal, Constant := false, ExternalAccess := Rea
TAG     Motor_Starter_Program   Jog_PB  "Pushbutton to jog the conveyor forward"        "BOOL"  ""      "(RADIX := Decimal, Constant := fals
TAG     Motor_Starter_Program   Language        "LD = 0$NFBD = 1$NSFC = 2$NST = 3"      "DINT"  ""      "(RADIX := Decimal, Constant := fals
TAG     Motor_Starter_Program   Motor_1_AuxContact      ""      "Simulate_Feedback"     ""      ""
TAG     Motor_Starter_Program   Motor_1_Clear_Fault_PB  ""      "BOOL"  ""      "(RADIX := Decimal, Constant := false, ExternalAccess := Rea
TAG     Motor_Starter_Program   Motor_Out_FBD   "Output command to the conveyor motor"  "BOOL"  ""      "(RADIX := Decimal, Constant := fals
TAG     Motor_Starter_Program   Motor_Out_LD    "Output command to the conveyor motor"  "BOOL"  ""      "(RADIX := Decimal, Constant := fals
TAG     Motor_Starter_Program   Motor_Out_SFC   "Output command to the conveyor motor"  "BOOL"  ""      "(RADIX := Decimal, Constant := fals
TAG     Motor_Starter_Program   Motor_Out_ST    "Output command to the conveyor motor"  "BOOL"  ""      "(RADIX := Decimal, Constant := fals
TAG     Motor_Starter_Program   Motor_Starter_FBD       "Conveyor"      "Motor_Starter" ""      ""
TAG     Motor_Starter_Program   Motor_Starter_LD        "Conveyor"      "Motor_Starter" ""      ""
TAG     Motor_Starter_Program   Motor_Starter_SFC       "Conveyor"      "Motor_Starter" ""      ""
TAG     Motor_Starter_Program   Motor_Starter_ST        "Conveyor"      "Motor_Starter" ""      ""
TAG     Motor_Starter_Program   Reset   ""      "BOOL"  ""      "(RADIX := Decimal, Constant := false, ExternalAccess := ReadOnly)"
TAG     Motor_Starter_Program   Start_PB        "Pushbutton to start the conveyor"      "BOOL"  ""      "(RADIX := Decimal, Constant := fals
TAG     Motor_Starter_Program   Step_000        ""      "SFC_STEP"      ""      ""
TAG     Motor_Starter_Program   Step_001        ""      "SFC_STEP"      ""      ""
TAG     Motor_Starter_Program   Step_002        ""      "SFC_STEP"      ""      ""
TAG     Motor_Starter_Program   Stop_PB "Pushbutton to stop the conveyor"       "BOOL"  ""      "(RADIX := Decimal, Constant := false, Exter
TAG     Motor_Starter_Program   Tran_000        ""      "BOOL"  ""      "(RADIX := Decimal, Constant := false, ExternalAccess := ReadOnly)"
TAG     Motor_Starter_Program   Tran_001        ""      "BOOL"  ""      "(RADIX := Decimal, Constant := false, ExternalAccess := ReadOnly)"
TAG     Motor_Starter_Program   Tran_002        ""      "BOOL"  ""      "(RADIX := Decimal, Constant := false, ExternalAccess := ReadOnly)"
TAG     Motor_Starter_Program   UseConveyorSimulation   ""      "BOOL"  ""      "(RADIX := Decimal, Constant := false, ExternalAccess := Rea
TAG     Motor_Starter_Program   Use_Real_I_O    ""      "BOOL"  ""      "(RADIX := Decimal, Constant := false, ExternalAccess := ReadOnly)"
TYPE    SCOPE   ROUTINE COMMENT OWNING_ELEMENT  LOCATION
TEXTBOX "Motor_Starter_Program" "Motor_Starter_FBD"     "Gives simple conveyor control such as start, stop, jog, and $R$Nfault detection"
""      60:20:Sheet1
RCOMMENT        "Motor_Starter_Program" "Motor_Starter_LD"      "If Job_PB = on, then jog the conveyor forward."        "OTE
(Motor_Starter_LD.Jog)" "0"
TEXTBOX "Motor_Starter_Program" "Motor_Starter_SFC"     "SFC Execution Configuration$N$NLast Scan of Active Steps = $R$NAutomatic Reset"
""      360:40
RCOMMENT        "Motor_Starter_Program" "Nested_Motor_Starter_LD"       "Conveyor Control$NThese rungs are an example of using the
Conveyor_Control Add-On Instruction."   "NOP()" "0"
```

# Considerations for using Microsoft Excel to edit a.CSV file

## Introduction

This appendix describes how using Windows or Excel to edit a.CSV file can affect the file.

**Important:** To edit the.CSV file, it is recommended that you use a database program tool, such as Access®, or a raw text editor. Many other desktop tools, such as Windows or Excel, might change the structure of the.CSV file and cause an import of the file to fail.

## Recommendations

To use Microsoft Excel to edit your.CSV tag file:

- Use single quotes instead of double quotes within descriptions and comments.
- Do not create descriptions or comments that consist only of numbers, have leading zeros, or have a leading symbol that Microsoft Excel treats specially. For example, do not create descriptions, such as:

  002
  +2
  =2
  -2
  .0

- Do not create descriptions or comments that start with a +, -, or = symbol. If you add text after the symbol, Excel displays #NAME? in the cell.

## Logix Designer data transformations

When Logix Designer application exports tags, it performs these conversions.

| Original Content | Content in.CSV File After Export |
|---|---|
| ‘ | $’ |
| " | $Q |
| newline | $N$L |
| tab | $T |
| $ | $$ |

## Microsoft Excel Data Transformation

When you open the exported.CSV file in Excel, Excel makes these conversions.

| Original Content | Content in.CSV File After Export | Content After Opening in Excel | Content After Saving from Excel | Details |
|---|---|---|---|---|
| .0 | ".0" | 0 | 0 | Logix Designer addresses this as the specifier for a tag.<br>If you enter this as an entire comment, you lose any preceding period (.). If you enter any text before or after this, Excel maintains the content. |
| =2 | "=2" | 2 | 2 | If you enter this as an entire comment, you lose any preceding equal sign (=). If you enter any text before or after this, Excel maintains the content. |
| +2 | "+2" | 2 | 2 | If you enter this as an entire comment, you lose any preceding plus sign (+). If you enter any text before or after this, Excel maintains the content. |
| 002 | "002" | 2 | 2 | If you enter this as an entire comment, you lose any preceding zeros. If you enter any text before or after this, Excel maintains the content. |
| test string | "test string" | test string | test string | Excel puts quotes around cell contents only if there is an embedded comma.<br>Logix Designer always places double quotes around text. But Logix Designer still can handle the description without quotes. |
| "test string" | "$"test string$"" | $test string$"" | "$test string$""""" | Excel and Logix Designer alter content when it includes a dollar sign ($). |
| has "quoted text" within string | "has $"quoted text$" within string" | has $quoted text$" within string" | "has $quoted text $"" within string""" | Excel and Logix Designer alter content when it includes a dollar sign ($). |
| this has ‘embedded’ text | this has $’embedded$’ text | this has $’embedded$’ text | this has $’embedded$’ text | Single quotes work in both software packages. |
| +text | "+text" | #NAME? | #NAME? | Do not start a description or comment with a plus sign (+). |
| -text | "-text" | #NAME? | #NAME? | Do not start a description or comment with a minus sign (-). |
| =text | "=text" | #NAME? | #NAME? | Do not start a description or comment with an equal sign (=). |

# Import/Export revision history

## Introduction

This appendix contains a history of enhancements made to the import/export feature since L5K version 1.1, major revision 1, and minor revision 1 that was included with Logix Designer, version 8.0.

These releases of the import/export feature L5K version correspond to these releases of Logix Designer and the Logix Designer application.

| RSLogix 5000 /Logix Designer Version | Import/Export L5K Version |
|---|---|
| 32.xx | 2.23 |
| 31.xx | 2.22 |
| 30.xx | 2.21 |
| 29.xx | 2.20 |
| 28.xx | 2.19 |
| 27.xx | 2.18 |
| 26.xx | 2.17 |
| 24.XX | 2.15 |
| 23.xx | 2.14 |
| 21.xx | 2.12 |
| 20.xx | 2.11 |
| 19.xx | 2.10 |
| 18.xx | 2.9 |
| 17.xx | 2.8 |
| 16.xx | 2.7 |
| 15.xx | 2.6 |
| 13.xx | 2.4 |
| 12.xx | 2.3 |
| 11.xx | 2.2 |
| 10.xx | 2.1 |
| 9.00 | 2.0 |
| 5.02 | 1.2 |
| 8.xx, 7.xx, 6.xx, 2.xx | 1.1 |
| 1.23, 1.21 | 1.0 |
| 1.11, 1.10 | 0.4 |

## Backward compatibility

The import/export feature supports backward compatibility for import operations. Therefore, the application can import.L5K or.L5X files that are generated by a previous version of the programming software. In some cases, an older.L5K file might not correctly import into a newer version of the application. The revision history in this appendix lists any conditions when backward compatibility for an import operation does not work as expected.

The import/export feature **does not** support backward compatibility for export operations. Therefore, older versions of the application cannot read.L5K or.L5X files that are created with newer versions of the application. In some cases, a.L5K or.L5X file created with a newer version of the application may import with warnings into an older version of the application. In these cases, attributes on components may be set to default values during import.

Each version of the application exports.L5K files with a specific import/export L5K version number. The application imports any.L5K file with the same major revision number and the same or lower minor revision number. The major L5K revision number increments when there are conditions such that the application cannot support backward compatibility for L5K import operations. The minor L5K revision number increments whenever there is a change in the file, such as a new module, an attribute is added, or the set of options for an attribute is changed, that does not affect backward compatibility for L5K import operations.

.L5X files use the XML open standard format. L5X files do not have a revision number associated with them.

**Important:** Use caution when copying and pasting components between different versions of the application. The Logix Designer only supports pasting to the same version or newer version of the application. Pasting to a prior version of RSLogix 5000 software is not supported. When pasting to a prior version, the paste action may succeed, but the results may not be what you expect.

## Import/Export version 2.23 Logix Designer application version 32

Version 2.23 of the Import/Export feature that is included with the Logix Designer application, version 32, includes the following major enhancement:

- Added the new FBD Function element.
- Added new axis tag attributes.

## Import/Export version 2.22 Logix Designer application version 31

Version 2.22 of the Import/Export feature that is included with the Logix Designer application, version 31, includes the following major enhancement:

- Added the new Alarm Condition type.

## Import/Export version 2.21 Logix Designer application version 30

Version 2.21 of the Import/Export feature that is included with the Logix Designer application, version 30, includes the following major enhancements:

- Added the TrackingGroups attribute for routines, Add-On Instructions, and tags.
- Added encryption elements and attributes for routines and Add-On Instructions.

## Import/Export version 2.20 Logix Designer application version 29

Version 2.20 of the Import/Export feature that is included with the Logix Designer application, version 29, includes the following major enhancement:

- Added the EtherNetIPMode controller attribute.

## Import/Export version 2.19 Logix Designer application version 28

Version 2.19 of the Import/Export feature that is included with the Logix Designer application, version 28, includes these major enhancements:

- Added Equipment Sequence functionality.
- Added the Primary Action Set controller security attribute.
- Added the Permission Set controller attribute.

## Import/Export version 2.18 Logix Designer application version 27

Version 2.18 of the Import/Export feature that is included with the Logix Designer application, version 27, includes these major enhancements:

- Added the custom properties functionality.
- Added the DownloadCustomProperties controller attribute.
- Added the HMI Button Control (HMIBC) instruction.

## Import/Export version 2.17 Logix Designer application version 26

Version 2.17, major revision 2, minor revision 17, of the Import/Export feature that is included with the Logix Designer application, does not include any major enhancements.

## Import/Export version 2.15 Logix Designer application version 24

Version 2.15, major revision 2, minor revision 15, of the Import/Export feature that is included with the Logix Designer application, version 24 includes these major enhancements:

- Removed support for the 1789-L60 controller.
- Added the ParameterConnections controller element with the EndPoint1 and EndPoint2 attributes.

- Added the SafetyEnabled and NATActualAddress module attributes.
- Added the SignatureID, SignatureTimestamp, SafetySignatureID Encoded Add-On Instruction attributes.
- Added the Sequencing tag attribute.
- Added the Alternate1UpdateMultiplier and Alternate2UpdateMultiplier motion group tag attributes.
- Added the AxisUpdateSchedule axis tag attribute.
- Added the UseAsFolder program attribute.
- Added the ChildPrograms program element with the child_program_name attribute.

There were no feature changes for the Import/Export feature for version 2.14, major revision 2, minor revision 14, included with the Logix Designer application, version 23.

## Import/Export version 2.12 Logix Designer application version 21

Version 2.12, major revision 2, minor revision 12, of the Import/Export feature included with Logix Designer application, version 21 includes these major enhancements:

- Removed support for the following controllers: 1756-L61, 1756-L61S, 1756-L62, 1756-L62S, 1756-L63, 1756-L63S, 1756-L64, 1756-L65, 1768-L43, 1768-L43S, 1768-L45, 1768-L45S, 1769-L23E-QBF1, 1769-L23E-QBFC1, 1769-L23-QBFC1, 1769-L31, 1769-L32C, 1769-L32E, 1769-L35CR, 1769-L35E
- Added the PassThroughConfiguration controller attribute.
- Added Engineering Unit, State0, State1, Max, and Min attributes to datatype and tag components and Add-On Instructions parameters and local tags.
- Added new attributes for analog alarm tags: HHOperShelve, HOperShelve, LOperShelve, LLOperShelve, ROCPosOperShelve, ROCNegOperShelve, ProgUnshelveAll, HHOperUnshelve, HOperUnshelve, LOperUnshelve, LLOperUnshelve, HHMinDurationEnable, HMinDurationEnable, LMinDurationEnable, LLMinDurationEnable, ROCPosOperUnshelve, ROCNegOperUnshelve, ShelveDuration, MaxShelveDuration.
- Added new attributes for digital alarm tags: OperShelve, ProgUnshelve, OperUnshelve, ShelveDuration, MaxShelveDuration

## Import/Export version 2.11 Logix Designer version 20

Version 2.11, major revision 2, minor revision 11, of the import/export feature included with Logix Designer, version 20 includes these major enhancements:

- Additional controllers supported.
- SecurityAuthorityID, SecurityAuthorityURI, ChangesToDetect, and Trusted Slots attributes added to the controller component.
- SignatureRunModeProtect attribute added to the safety controller system.
- ShutdownParentOnFault, DrivesADCMode, DrivesADCEnabled, and UserDefinedCatalogNumber added to the module component.
- ConfigScript sub section added to the L5K and L5X formats of the module component.
- Priority, InputConnectionType, OutputRedundantOwner, InputProductionTrigger, ConnectionPath, InputTagSuffix, and OutputTagSuffix attributes added to the module connection component.
- LargePackageUsage attributed added for message tags.
- MasterInputConfigurationBits and MasterPositionFilterBandwidth attributes added for coordinate system tags.
- AdditionalBusCapacitance and InterpolatedPositionConfiguration attributes added for axis tags.
- Internet Protocol, Ethernet Ports, and Ethernet Network controller configuration objects added for controllers that have embedded Ethernet ports in them.

## Import/Export version 2.10 Logix Designer version 19

Version 2.10 (major revision 2, minor revision 10) of the import/export feature included with Logix Designer, version 19 includes these major enhancements:

- Procedures for configuring source-protected components in encrypted or clear text format.
- ControlLogix 1756-L73 and 1756-L75 controllers added to the list of processor types.

## Import/Export version 2.8 Logix Designer version 18

Version 2.8, major revision 2, minor revision 9, of the import/export feature included with Logix Designer, version 18 includes these major enhancements:

- Addition of ControlLogix 1756-L73 and 1756-L75 controller types and Compact GuardLogix®1768-L43S and 1768-L45S controller types.
- Addition of CanUseRPIFromController attribute to the Controller component.
- Addition of the WatchList element to a Controller declaration.

- Addition of TagType and AliasFor parameters to the L5X format of the Add-On Instruction definition
- Addition of safety abilities to Add-On Instructions.
- Addition of the External Access and Constant attributes to Tag components.
- Addition of new attributes to axis tags,

This version of import/export also supports the AXIS_CIP_DRIVE tag. For information, see the Integrated Motion on the Ethernet/IP Network Configuration and Startup User Manual, publication MOTION-UM003.

- Addition of MinimumRPI, MaximumRPI, and DefaulRPI attributes to ta produced Tag component.
- Addition of attributes to support unicast communication for I/O modules on EtherNet/IP networks:
  - Unicast attribute added to the Connection element of the Module component.
  - Unicast attribute added to the SafetyProducedTag component.
  - UnicastPermitted attribute added to the SafetyConsumedTag component.

## Import/Export version 2.8 Logix Designer version 17

Version 2.8, major revision 2, minor revision 8, of the import/export feature included with Logix Designer, version 17 includes these major enhancements:

- 1756-L63S GuardLogix® safety controller and safety relay ladder instructions.
- 1756-L65, 1768-L45, 1769-L23E-QB1, 1769-L23E-QBFC1, 1769-L23-QBFC1 controllers.
- A tag *IncludeConnectionStatus* attribute is no longer exported.
- The L5X format for rung export has been modified such that rung UIDs are no longer included in the export format.

## Import/Export version 2.7 Logix Designer version 16

Version 2.7, major revision 2, minor revision 7, of the import/export feature included with Logix Designer, version 16 includes these major enhancements:

- 1756-L61S and 1756-L62S GuardLogix safety controllers and safety relay ladder instructions.
- 1756-L64 ControlLogix controller.
- Updated CONTROLLER example.

- Add-On Instructions.
- Alarms
  - New alarm instructions: ALMA, ALMD.
  - Digital and analog alarm tags.
- New instructions
  - Motion instructions: MCT, MCTP.
  - Safety instructions: DIN, RIN, ESTOP, ENPEN, LC, FPMS, ROUT, THRS.
- Addition of ShareUnusedTimeSlice and InhibitAutomaticFirmwareUpdate attributes to the CONTROLLER component.
- Addition of UserDefinedVendor, UserDefinedProductType, userDefinedProductCode, UserDefinedMajor, and UserDefinedMinor attributes to the MODULE component.
- Addition of LINT data type.
- Addition of Unicast and UnicastPermitted attributes to the TAG component.
- Additional attributes and valid values for existing attributes to AXIS tags.
- Additional attributes for COORDINATE_SYSTEM tags.
- Source protected routines and Add-On-Instructions appear as encrypted data in export files. In previous releases, source protected data was not exported at all.
- Addition of SynchronizeRedundancyDataAfterExecution attribute to the PROGRAM component.
- Additional CONFIG attributes.
- New export TXT format for rungs and logic comments that uses tabs to separate values. This format is similar to the CSV format that uses commas to separate values.

  The CSV and TXT formats also include text box comments from function block and sequential function chart logic.

## Import/Export version 2.6 Logix Designer version 15

Version 2.6, major revision 2, minor revision 6, of the import/export feature included with Logix Designer, version 15 includes these major enhancements:

- Support for the 1769-L32C, 1769-L32CR CompactLogix and 1768-L43 CompactLogix controllers.

This release also removed support for the 1756-L1 CompactLogix, 1794-L33 FlexLogix, 1769-L20 CompactLogix, 1769-L30 CompactLogix, and PowerFlex 700 S controllers.

- Equipment Phase program type and its relay ladder and structured text instructions.
- ControlLogix and SoftLogix controllers now support 100 programs per task.
- Information about when an imported file modifies a project so that you cannot go online and access a previously downloaded controller.
- Additional values for the Mode attribute of a MODULE component.
- New SERCOS IDN Read and SERCOS IDN Write message types.
- New motion AXIS_GENERIC_DRIVE type.
- Removal of the DescriptionWidth parameter from the STEP, TRANSITION, and STOP components in SFC logic.
- Addition of an Attributes column to the CSV format for exported tags.

## Import/Export version 2.4 Logix Designer version 13

Version 2.4, major revision 2, minor revision 4, of the import/export feature included with Logix Designer, version 13 includes these major enhancements:

- Support for new controllers.
- ExtendedProp section to MODULE data.
- Support for new TAG attributes.

  Attributes can be in any order in an import/export file. The order shown in this document is the order the attributes export.
- Support for a TREND object in the import/export.L5K file.
- New MCSV instruction in ladder logic (chapter 4) and structured text.
- Online editing support for structured text and sequential function chart logic.
- Updated CSV format now includes rung comments.
- New L5X format for partial import/export of ladder rungs, tags, and trends.

## Import/Export version 2.3 Logix Designer version 12

Version 2.3, major revision 2, minor revision 3, of the import/export feature included with Logix Designer, version 12.01 includes these major enhancements:

- The structured text component changed from STX_ROUTINE to ST_ROUTINE. The LanguageType attribute in SFC routines for embedded structured text also changed from STX to ST.
- Support for new controllers.

- Addition of the ControlNetSignature attribute to the MODULE component.
- Addition of the ProgrammaticallySendEventTrigger attribute to the TAG component.
- New COORDINATE_SYSTEM tag.
- Addition of several new attributes to the axis tag types.
- Addition of DisableFlag attribute to the PROGRAM component.
- Addition of EventTrigger and EventTag attributes to the TASK component to support Event tasks.
- New EVENT, IOT, MCCD, MCCM, MCLM, MCS, MCSD, and MCSR instructions in ladder logic and structured text.
- Addition of information regarding the LOGIC block when exporting online function block logic.
- Addition of new modules and their valid CommMethod and ConfigMethod values.

## Import/Export version 2.2 Logix Designer version 11

Version 2.2, major revision 2, minor revision 2, of the import/export feature included with Logix Designer, version 11.10 includes these major enhancements:

- Support for the 1756-L63 controller.
- New controller attributes to support sequential function charts.
- Corrected the DATATYPE attributes and added the FamilyType attribute.
- Additional information for the CompatibleModule and KeyMask attributes of the MODULE component.
- Addition of RSNetWorxFileName attribute to the MODULE component.
- Addition of SFC_ACTION, SFC_STEP, and SFC_STOP tag types.
- Addition of 38400 as a supported serial port baud rate.
- Addition of structured text instructions.
- Addition of EOT, SFR, and SFP instructions to relay ladder and structured text.
- Addition of sequential function chart components.
- Addition of an appendix that lists the valid CommMethod and ConfigMethod values for the supported I/O modules.

Beginning with version 2.2, multi-line rung comments with hard returns are no longer exported as one long string in double-quotes. Instead, each line of a multi-line rung comment is on a separate line in the.L5K file with double-quotes around each line. When imported, the multiple quoted strings are concatenated to form

the rung comment. This improves the readability of the.L5K text file by using the existing multiple-string capability of the rung comment syntax. Older formats still work on import.

## Import/Export version 2.1 Logix Designer version 10

Version 2.1, major revision 2, minor revision 1, of the import/export feature included with Logix Designer, version 10.0 includes these major enhancements:

- Removal of the characters when specifying a controller type.
- Addition of the SecurityCode attribute to the Controller object.
- Enhancements to the Message tag structure. See page 264.
- The Program object now includes a Mode attribute.
- Correction to valid values for Watchdog and Rate attributes of the Task object.
- Addition of MaxStationAddress and TokenHoldFactor attributes to the Config DF1 object.
- Addition of new instructions: SIZE, SWPB, LOWER, and UPPER.
- The NumberOfAppendChars of the Config ASCII object is no longer exported. If you have an import/export file with any of these attributes, the file will correctly import into the software. This attributes will be removed when you export the file.

## Changes to support MESSAGE tag enhancements

Version 2.1 (major revision 2, minor revision 1) of the import/export feature that is included with Logix Designer 2 programming software, version 10.0 made significant changes to the MESSAGE tag. For reference, this table shows the MESSAGE tag structure of the previous import/export release.

**MESSAGE Tag Structure (Version 2.0)**

| Attribute | Description |
|---|---|
| Description | Provide information about the tag.<br>Specify Description := "*text*" |
| Comment | Provide information about a tag component.<br>Specify Comment<*specifier*> := "*text*"<br>Where the *specifier* is:<br>.*bitnumber*for a bit in the tag<br>[*element*]for an array element of the tag<br>.*membername*for a structure member of the tag |
| MessageType | Type **Block Transfer Read**, **Block Transfer Write**, **CIP Data Table Read**, **CIP Data Table Write, CIP Generic**, **PLC2 Unprotected Read, PLC2 Unprotected Write**, **PLC3 Typed Read**, **PLC3 Typed Write**, **PLC3 Word Range Read**, **PLC3 Word Range Write**, **PLC5 Typed Read**, **PLC5 Typed Write**, **PLC5 Word Range Read**, **PLC5 Word Range Write**, **SLC Typed Read**, or **SLC Typed Write**.<br>Specify MessageType := *text* |

| Attribute | Description |
|---|---|
| RequestedLength | Specify the number of elements in the message instruction (0...32,767).<br>Specify RequestedLength := *value* |
| ConnectionPath | Specify the connection path to the other device.<br>Specify ConnectionPath := *string* |
| DF1DHFlag | If the communication method uses DH+, type 1. If the communication method does not use DH+, type 0.<br>Specify DF1DHFlag := *value* |
| LocalTag | Specify the tag name of the element in the local device.<br>Specify LocalTag := *text* |
| RemoteElement | Specify the tag name of the element in the remote device.<br>Specify RemoteElement := *value* |
| DHPlusSourceLink | If the communication method uses DH+, specify the source link (0...65,535).<br>Specify DHPlusSourceLink := *value* |
| DHPlusDestinationLink | If the communication method uses DH+, specify the destination link (0...65,535).<br>Specify DHPlusDestinationLink := *value* |
| DHPlusDestinationNode | If the communication method uses DH+, specify the destination node number (0...63 octal).<br>Specify DHPlusDestinationNode := *value* |
| DHPlusChannel | If the communication method uses DH+, specify the DH+ channel. Type **A** or **B**.<br>Specify DHPlusChannel := *letter* |
| CacheConnections | If the message is to cache connections, type **TRUE**. If the message is not to cache connections, type **FALSE**.<br>Specify CacheConnections := *text* |
| ServiceCode | If the message type is CIP Generic, specify the service code (0...255 hexadecimal).<br>Specify ServiceCode := #16*value* |
| ObjectType | If the message type is CIP Generic, specify the object type (0...65,535 hexadecimal).<br>Specify ObjectType := *16#value* |
| TargetObject | If the message type is CIP Generic, specify the target object (0...65,535 decimal).<br>Specify TargetObject := *value* |
| AttributeNumber | If the message type is CIP Generic, specify the attribute number (0...65,535 hexadecimal).<br>Specify AttributeNumber := *16#value* |
| DestinationTag | Specify the tag name of the destination element.<br>Specify DestinationTag := *text* |

## Import/Export version 2.0 Logix Designer version 9

Version 2.0 (major revision 2, minor revision 0) of the import/export feature included with Logix Designer, version 9.0 includes these major enhancements:

- Replaced the AXIS tag with AXIS_CONSUMED, AXIS_SERVO, AXIS_SERVO_DRIVE, and AXIS_VIRTUAL tags.
- For any attribute that you can specify a *not applicable* state, type **<NA>**, rather than NA.
- Revised manual that includes a description and example of the STRING data type.

**Important:** Version 9 of Logix Designer only supports ControlLogix processors.

## Motion changes to support the SERCOS Protocol

Version 2.0, major revision 2, minor revision 0, of the import/export feature included with Logix Designer, version 9.0 includes significant changes to motion-related tags to support the SERCOS protocol.

- CoarseUpdatePeriod and AutoTagUpdate parameters were added to the MOTION_GROUP tag to support SERCOS. For reference, the previous structure is described on page 267 on page 288.
- Earlier versions of the import/export feature supported one AXIS tag. To support SERCOS, the import/export feature replaced AXIS with four axis tags: AXIS_CONSUMED, AXIS_SERVO, AXIS_SERVO_DRIVE, and AXIS_VIRTUAL. The previous AXIS tag is incorporated into these new tags, but no longer exists as its own tag. For reference, the AXIS structure is described on page 267 on page 289.

  If you have a version 8.0 import/export file with AXIS tags that you import into version 9.0 software, after changing the import/export version line to 2.0), the AXIS tags convert to:

| If the AXIS type is: | It Converts to: |
|---|---|
| Unused | AXIS_SERVO |
| Position only | AXIS_SERVO |
| Servo | AXIS_SERVO |
| Consumed | AXIS_CONSUMED |
| Virtual | AXIS_VIRTUAL |

## MOTION_GROUP tag structure (version 1.1)

| Attribute | Description |
|---|---|
| Description | Provide information about the tag.<br>Specify Description := "*text*" |
| Comment | Provide information about a tag component.<br>Specify Comment<*specifier*> := "*text*"<br>Where the *specifier* is:<br>.*bitnumber*for a bit in the tag<br>[*element*]for an array element of the tag<br>.*membername*for a structure member of the tag |
| GroupType | Specify the type of motion group, such as Independent.<br>Specify GroupType := *text* |

| Attribute | Description |
|---|---|
| CoarseUpdateMultiplier | Specify the coarse update rate (5-320ms).<br>Specify CoarseUpdateMultiplier := *value* |
| ServoUpdatePeriod | Specify the servo update period in milliseconds (any positive number)<br>Specify ServoUpdatePeriod := *value* |
| PhaseShift | Specify the phase shift (0-65,535).<br>Specify PhaseShift := *value* |
| GeneralFaultType | Specify whether an error generates a major fault or a non-major fault. Type **Major Fault** or **Non Major Fault**.<br>Specify GeneralFaultType := *text* |

## AXIS tag structure (version 1.1)

**AXIS tag attributes (version 1.1)**

| Attribute | Description |
|---|---|
| Description | Provide information about the tag.<br>Specify Description := "*text*" |
| Comment | Provide information about a tag component.<br>Specify Comment<*specifier*> := "*text*"<br>Where the *specifier* is:<br>.*bitnumber*for a bit in the tag<br>[*element*]for an array element of the tag<br>.*membername*for a structure member of the tag |
| MotionGroup | Type the name of the associated motion group, or type **NA**.<br>Specify MotionGroup := *text* |
| MotionModule | Type the name of the associated motion module, or type **NA**.<br>Specify MotionModule := *text* |
| AxisState | Type **Axis-Ready**, **Direct Drive Control**, **Servo Control**, **Axis Faulted**, or **Axis Shutdown**.<br>Specify AxisState := *text* |
| PositionUnits | Specify the type of units.<br>Specify PositionUnits := *text* |
| TimeUnits | Type **Seconds** or **Minutes**.<br>Specify TimeUnits := *text* |
| InstructionSpeedUnits | Type **Percentage** or **Engineering Units**.<br>Specify InstructionSpeedUnits := *text* |
| InstructionAccelDecelUnits | Type **Percentage** or **Engineering Units**.<br>Specify InstructionAccelDecelUnits := *text* |
| InstructionMoveProfile | Type **Trapezoidal** or **S-Curve**.<br>Specify InstructionMoveProfile := *text* |
| InstructionJogProfile | Specify **Trapezoidal** or **S-Curve**.<br>Specify InstructionJogProfile := *text* |
| ConversionConstant | Specify the conversion constant. Type a real number from 1.0...1.0e9.<br>Specify ConversionConstant := *value* |
| HomeMode | Type **Passive** or **Active**.<br>Specify HomeMode := *text* |

| Attribute | Description |
|---|---|
| HomeSequenceType | Type **Immediate Home**, **Home To Switch**, **Home To Marker Only**, or **Home To Switch With Marker**.<br>Specify HomeSequenceType := *text* |
| HomePosition | Specify the home position (any positive number).<br>Specify HomePosition := *value* |
| HomeSpeed | Specify the home speed (any positive number).<br>Specify HomeSpeed := *value* |
| HomeReturnSpeed | Specify the home return speed (any positive number).<br>Specify HomeReturnSpeed := *value* |
| MaximumSpeed | Specify the maximum speed (any positive number).<br>Specify MaximumSpeed := *value* |
| MaximumAcceleration | Specify the maximum acceleration (any positive number).<br>Specify MaximumAcceleration := *value* |
| MaximumDeceleration | Specify the maximum deceleration (any positive number).<br>Specify MaximumDeceleration := *value* |
| ProgrammedStopMode | Type **Fast Stop**, **Fast Shutdown**, or **Hard Shutdown**.<br>Specify ProgrammedStopMode := *text* |
| AverageVelocityTimebase | Specify the average velocity timebase (any positive number).<br>Specify AverageVelocityTimebase := *value* |
| ServoStatusUpdateBits | Specify the servo status update bits. Type a hexadecimal number.<br>Specify ServoStatusUpdateBits := 16#*value* |
| MotionConfigurationBits | Specify the motion configuration bits. Type a hexadecimal number.<br>Specify MotionConfigurationBits := 16#*value* |
| AxisType | Type **Unused**, **Position Only**, **Servo**, **Consumed**, or **Virtual**.<br>Specify AxisType := *text* |
| PositionUnwind | Specify the unwind position (0-65,535).<br>Specify PositionUnwind := *value* |
| MaximumPositiveTravel | Specify the maximum positive travel (any positive number).<br>Specify MaximumPositiveTravel := *value* |
| MaximumNegativeTravel | Specify the maximum negative travel (any positive number).<br>Specify MaximumNegativeTravel := *value* |
| PositionErrorTolerance | Specify the position error tolerance (any positive number).<br>Specify PositionErrorTolerance := *value* |
| PositionLockTolerance | Specify the position local tolerance (any positive number).<br>Specify PositionLockTolerance := *value* |
| PositionProportionalGain | Specify position proportional gain (any positive number).<br>Specify PositionProportionalGain := *value* |
| PositionIntegralGain | Specify the position integral gain (any positive number).<br>Specify PositionIntegralGain := *value* |
| VelocityFeedforwardGain | Specify the velocity feedforward gain (any positive number).<br>Specify VelocityFeedforwardGain := *value* |
| AcclerationFeedforwardGain | Specify the acceleration feedforward gain (any positive number).<br>Specify AccelerationFeedforwardGain := *value* |
| VelocityProportionalGain | Specify the velocity proportional gain (any positive number).<br>Specify VelocityProportionalGain := *value* |

| Attribute | Description |
|---|---|
| VelocityIntegralGain | Specify velocity integral gain (any positive number).<br>Specify VelocityIntegralGain := *value* |
| OutputFilterBandwidth | Specify output filter bandwidth (any positive number).<br>Specify OutputFilterBandwidth := *value* |
| OutputScaling | Specify the output scaling (any positive number).<br>Specify OutputScaling := *value* |
| OutputLimit | Specify the output limit (any positive number).<br>Specify OutputLimit := *value* |
| OutputOffset | Specify output offset (any positive number).<br>Specify OutputOffset := *value* |
| FrictionCompensation | Specify friction compensation (any positive number).<br>Specify FrictionCompensation := *value* |
| SoftOvertravelFaultAction | Type **Shutdown**, **Disable Drive**, **Stop Motion**, or **Status Only**.<br>Specify SoftOvertravelFaultAction := *text* |
| PositionErrorFaultAction | Type **Shutdown**, **Disable Drive**, **Stop Motion**, or **Status Only**.<br>Specify PositionErrorFaultAction := *text* |
| EncoderLossFaultAction | Type **Shutdown**, **Disable Drive**, **Stop Motion**, or **Status Only**.<br>Specify EncoderLossFaultAction := *text* |
| EncoderNoiseFaultAction | Type **Shutdown**, **Disable Drive**, **Stop Motion**, or **Status Only**.<br>Specify EncoderNoiseFaultAction := *text* |
| DriveFaultAction | Type **Shutdown**, **Disable Drive**, **Stop Motion**, or **Status Only**.<br>Specify DriveFaultAction := *text* |
| ServoConfigurationBits | Specify the servo configuration bits. Type a hexadecimal number.<br>Specify ServoConfigurationBits := 16#*value* |
| MotorEncoderTestIncrement | Specify the motor encoder test increment (any positive number).<br>Specify MotorEncoderTestIncrement := *value* |
| TuningTravelLimit | Specify the tuning travel limit (any positive number).<br>Specify TuningTravelLimit := *value* |
| TuningSpeed | Specify the tuning speed (any positive number).<br>Specify TuningSpeed := *value* |
| DampingFactor | Specify the damping factor (any positive number).<br>Specify DampingFactor := *value* |
| PositionServoBandwidth | Specify position servo bandwidth (any positive number).<br>Specify PositionServoBandwidth := *value* |
| TuningConfigurationBits | Specify the tuning configuration bits. Type a hexadecimal number.<br>Specify TuningConfigurationBits := 16#*value* |

## Import/Export version 1.1 Logix Designer version 8

Version 1.1, major revision 1, minor revision 1, of the import/export feature included with Logix Designer, version 8.0 includes these major enhancements:

- Function block instructions and routines.
- ASCII instructions.

- Verification of all instruction attributes and parameters.

## A


action
- attributes 213
- L5K structure 213
- L5X structure 213

Add-On Instruction
- attributes 193
- attributes in FBD 193
- component 89
- encoded data 99
- guidelines 102
- in a function block diagram routine 193
- in FBD guidelines 102
- L5K example 102
- L5K source-protected example 101
- L5K structure 90
- L5K structure in FBD 193
- L5X example 102
- L5X source-protected example 100
- L5X structure 89
- L5X structure in FBD 193
- local tag 97, 98
- local tag attributes 98
- parameter 92, 93, 97
- parameter attributes 95, 97
- routines 91
- source-protection 99
- tag values 140

alarm analog tag
- attributes 115
- L5K example 142
- L5K structure 115
- L5X example 142
- L5X structure 115

alarm digital tag
- attributes 118
- L5K example 142
- L5K structure 118
- L5X example 142
- L5X structure 117

AlarmConfig
- L5X structure 119

ALMMSG
- record 277

analog alarm message
- L5K structure 120

attachment
- attributes 197
- guidelines 197
- L5K structure 197
- L5X structure 197

attributes
- action 213
- Add-On Instruction 193
- Add-On Instruction in FBD 193
- alarm analog tag 115
- alarm digital tag 118
- attachment 197
- block 191
- branch 216
- child programs 167
- config objects 265
- consumed tag 114
- controller 53
- controller config objects 265
- coordinate system tag 136
- datatype 67
- datatype member 69
- directed link 217
- Encrypted content 100
- Encryption Key 100
- equipment phase 165
- function 193
- function block diagram routine 186
- icon 191
- iref 190
- jsr 194
- ladder logic routine 173
- leg 217
- local tag 98
- message tag 121
- module 74
- module connection 79
- motion group tag 138
- ocon 191
- oref 190
- parameter 95, 97
- parameter connections 251
- pen 259
- produced tag 114
- program 164

quick watch list 263
ret 195
rung 175
safety controller 56, 80
safety module 77
safety tag 139
sbr 195
sbrret 215
sequential function chart routine 208
sheet 187
step 211
stop 216
structured text routine 228
tag 112
task 248
text box 197
TrackingGroups 90, 112, 173, 186, 208, 228
transition 214
trend 255
watch tag 263
wire 196
axis tag
L5K structure 122
L5X structure 122


## B


backward compatibility 286
block
attributes 191
guidelines 192
L5K structure 191
L5X structure 191
boolean values
L5X format 42
branch
attributes 216
L5K structure 175
L5X structure 175


## C


child programs
attributes 167
components 167
guidelines 168
L5K example 170
L5K structure 167
L5X example 170
L5X structure 167
COMMENT
record 275
comments 30, 32, 49
CSV format 273
internal L5K file 49
internal L5X file 49
TXT Unicode format 273
complete import/export 25, 26, 28, 29
components
Add-On Instruction 89
basic L5K format 47
basic L5X format 39
child programs 167
config 265
datatype 67
descriptions in L5K 49
descriptions in L5X 49
display style in L5K 49
display style in L5X 43
function block diagram routine 185
ladder logic routine 173
module 73
parameter connections 251
program 163
quick watch list 263
sequential function chart routine 207
structured text routine 227
tag 109
task 247
trend 255
condition
L5K structure 215
L5X structure 215
config
component 265
L5K examples 265
L5X examples 265
config objects
attributes 265
L5K structure 265
L5X structure 265
consumed tag
attributes 114

controller
attributes 53
guidelines 57
L5K example 57
L5K safety example 57
L5K structure 52
L5X example 57
L5X safety example 57
L5X structure 51
safety attributes 56
controller config objects 265
attributes 265
L5K examples 265
L5X examples 265
coordinate system tag
attributes 136
L5K structure 136
L5X structure 136
CSV
ALMMSG 277
comment record 275
exporting 30
importing 32
TAG 273
CSV format 273, 283
examples 277

## D

data formats
L5X format 43
datatype
attributes 67
bit members 69
component 67
guidelines 71
L5K example 71
L5K structure 67
L5X example 71
L5X structure 67
member attributes 69
members 68
datatype member
attributes 69
bit members 69
L5K structure 68
L5X structure 68
descriptions
L5K format 49
L5X format 49
directed link
attributes 217
guidelines 217
L5K structure 217
L5X structure 217
display style
L5K format 49
L5X format 43

## E

encoded data
L5K example 101, 169
L5K structure 99
L5X example 100, 170
L5X structure 99
equipment phase
attributes 165
L5K example 170
L5X example 170
examples
CSV files 277
L5K ADD_ON_INSTRUCTION_DEFINITION 102
L5K ALARM_ANALOG tag 142
L5K ALARM_DIGITAL tag 142
L5K CHILD_PROGRAMS 170
L5K CONFIG 265
L5K CONTROLLER 57
L5K DATATYPE 71
L5K ENCODED_DATA 101, 169
L5K equipment phase PROGRAM 170
L5K FBD_ROUTINE 197
L5K LOGIC 188, 209, 229
L5K MODULE 81
L5K PROGRAM 170
L5K QUICK_WATCH_LIST 263
L5K ROUTINE 176
L5K safety CONTROLLER 57
L5K safety partner MODULE 81
L5K safety TAG 142
L5K safety TASK 248
L5K SFC_ROUTINE 197
L5K source-protected Add-On Instruction 101
L5K source-protected routine 169

L5K ST_ROUTINE 229
L5K TAG 142
L5K TASK 248
L5K TREND 259
L5X AddOnInstructionDefinition 102
L5X alarm analog Tag 142
L5X alarm digital Tag 142
L5X ChildPrograms 170
L5X Controller 57
L5X controller config objects 265
L5X DataType 71
L5X EncodedData 100, 170
L5X equipment phase Program 170
L5X FBDContent 188
L5X function block diagram Routine 197
L5X Module 81
L5X Program 170
L5X QuickWatchList 263
L5X Routine 175
L5X safety Controller 57
L5X safety partner Module 81
L5X safety Tag 142
L5X sequential function chart Routine 197
L5X SFCContent 209
L5X source-protected Add-On Instruction 100
L5X source-protected routine 170
L5X STContent 229
L5X structured text Routine 229
L5X Tag 142
L5X Task 248
L5X Trend 259
parameter connections 252
TXT Unicode files 277
Excel 283
exporting
comments 30
complete project 25, 28
CSV format 30
L5K format 25
L5X format 28
project 25, 28
tags 30
text boxes 30
TXT format 30


## F


format
CSV 273
TXT Unicode 273
function block diagram
instructions 200
neutral text 200
function block diagram routine
Add-On Instruction attributes 193
attachment attributes 197
attributes 186
component 185
FBDContent 188
jsr attributes 194
L5K example 188, 197
L5K online edits example 188
L5K sheet structure 187
L5K structure 185
L5X example 188, 197
L5X online edits example 188
L5X sheet structure 186
L5X structure 185
LOGIC 188
online edits 188
ret attributes 195
sbr attributes 195
sheet attributes 187
sheet guidelines 188
text box attributes 197
wire attributes 196
function block routine
block attributes 191
icon attributes 191
iref attributes 190
ocon attributes 191
oref attributes 190
functions 192


## G


guidelines
Add-On Instruction 102
Add-On Instruction in FBD 102
attachment 197
block 192
child programs 168

controller 57
datatype 71
directed link 217
icon 191
iref 190
jsr 195
module 81
ocon 191
oref 190
parameter connections 251
program 170
ret 196
rung 175
sbr 195
sheet 188
tag 142
task 248
text box 218
trend 259
wire 196

## H

history, import/export feature 285

## I

icon
attributes 191
guidelines 191
L5K structure 191
L5X structure 190
importing
comments 32
complete project 26, 29
CSV format 32
L5K format 26
L5X format 29
project 26, 29
tags 32
text boxes 32
TXT format 32
initial values 140
Add-On Instruction 140
L5K structure 140
L5X structure 140
instructions
function block diagram 200
ladder logic 177
structured text 231
internal file comments 49
iref
attributes 190
guidelines 190
L5K structure 190
L5X structure 190

## J

jsr
attributes 194
guidelines 195
L5K structure 194
L5X structure 194

## L

L5K
exporting 25
file conventions 48
file structure 47
importing 26
internal file comments 49
L5K structure
action 213
Add-On Instruction 90
Add-On Instruction in FBD 193
alarm analog tag 115
alarm digital tag 118
analog alarm message 120
attachment 197
axis tag 122
block 191
branch 175
child programs 167
condition 215
config objects 265
controller 52
coordinate system tag 136
datatype 67
datatype member 68
directed link 217
encoded data 99
function block diagram routine 185

icon 191
iref 190
jsr 194
ladder logic routine 173
limithigh 212
limitlow 212
local tag 98
message tag 121
module 73
module connection 78
motion group tag 138
ocon 191
oref 190
parameter 93, 97
parameter connections 251
pen 259
preset 212
program 164
quick watch list 263
ret 195
rung 174
rung branch 175
safety tag 139
sbr 195
sbrret 215
sequential function chart routine 207
sheet 187
source-protected Add-On Instruction 99
source-protected routine 99
step 211
stop 216
structured text routine 227
tag 110
tag initial values 140
task 247
text box 197
transition 214
trend 255
wire 196

L5X
boolean attribute values 42
data formats 43
exporting 28
file conventions 42
file structure 39
importing 29
internal file comments 49

L5X structure
action 213
Add-On Instruction 89
Add-On Instruction in FBD 193
alarm analog tag 115
alarm digital tag 117
AlarmConfig 119
attachment 197
axis tag 122
block 191
branch 175
child programs 167
condition 215
config objects 265
controller 51
coordinate system tag 136
datatype 67
datatype member 68
directed link 217
encoded data 99
function block diagram routine 185
icon 190
iref 190
jsr 194
ladder logic routine 173
limithigh 212
limitlow 213
local tag 97
message tag 120
module 73
module connection 78
motion group tag 138
ocon 190
oref 190
parameter 92, 97
parameter connections 251
pen 258
preset 212
program 163
quick watch list 263
ret 195
rung 174
rung branch 175
safety tag 139
sbr 195

sbrret 215
sequential function chart routine 207
sheet 186
source-protected Add-On Instruction 99
source-protected routine 99
step 210
stop 216
structured text routine 227
tag 109
tag initial values 140
task 247
text box 196
transition 214
trend 255
wire 196
ladder logic
instructions 177
neutral text 177
rung attributes 175
ladder logic routine
attributes 173
component 173
L5K example 176
L5K rung branch structure 175
L5K rung structure 174
L5K structure 173
L5X example 175
L5X rung branch structure 175
L5X rung structure 174
L5X structure 173
rung guidelines 175
language switching 50
leg
attributes 217
limithigh
L5K structure 212
L5X structure 212
limitlow
L5K structure 212
L5X structure 213
local tag
attributes 98
L5K structure 98
L5X structure 97


## M


message tag
attributes 121
L5K structure 121
L5X structure 120
Microsoft Excel 283
module
attributes 74
component 73
guidelines 81
L5K example 81
L5K safety partner example 81
L5K structure 73
L5X example 81
L5X safety example 81
L5X structure 73
safety attributes 77
module connection
attributes 79
L5K structure 78
L5X structure 78
safety attributes 80
motion group tag
attributes 138
L5K structure 138
L5X structure 138


## N


neutral text
function block diagram 200
ladder logic 177
structured text 231


## O


objects 265
ocon
attributes 191
guidelines 191
L5K structure 191
L5X structure 190
online edits 188, 209, 229
oref
attributes 190
guidelines 190

L5K structure 190
L5X structure 190

## P

parameter
attributes 95, 97
L5K structure 93, 97
L5X structure 92, 97
parameter connections
attributes 251
components 251
guidelines 251
L5K example 252
L5K structure 251
L5X example 252
L5X structure 251
pen
attributes 259
L5K structure 259
L5X structure 258
preset
L5K structure 212
L5X structure 212
produced tag
attributes 114
program
attributes 164
component 163
equipment phase attributes 165
guidelines 170
L5K equipment phase example 170
L5K example 170
L5K structure 164
L5X equipment phase example 170
L5X example 170
L5X structure 163
routine encoded data 99
source-protected routine 99
projects 25, 26, 28, 29

## Q

quick watch list
attributes 263
component 263
L5K example 263
L5K structure 263
L5X example 263
L5X structure 263
watch tag attributes 263

## R

RCOMMENT 273
record 275
Remark 273
ret
attributes 195
guidelines 196
L5K structure 195
L5X structure 195
routine
component 173, 185, 207, 227
encoded data 99
in Add-On Instruction 91
L5K example 176, 197
L5K source-protected example 169
L5X example 175, 197, 229
L5X source-protected example 170
source-protection 99
rung
attributes 175
guidelines 175
L5K structure 174
L5X structure 174
rung branch
L5K structure 175
L5X structure 175

## S

safety controller
attributes 56, 80
L5K example 57
L5K module example 81
L5K safety example 248
L5K tag example 142
L5X example 57
L5X module example 81
L5X tag example 142
module attributes 77
safety partner module
L5K example 81

L5X example 81
safety tag
attributes 139
L5K example 142
L5K structure 139
L5X example 142
L5X structure 139
sbr
attributes 195
guidelines 195
L5K structure 195
L5X structure 195
sbrret
attributes 215
L5K structure 215
L5X structure 215
sequential function block routine
L5K example 209
L5K online edits example 209
sequential function chart
attachment attributes 197
attachment guidelines 197
directed link guidelines 217
text box attributes 197
text box guidelines 218
sequential function chart routine
action attributes 213
attributes 208
branch attributes 216
component 207
directed link attributes 217
L5K action structure 213
L5K attachment structure 197
L5K branch structure 175
L5K condition structure 215
L5K directed link structure 217
L5K example 197
L5K limithigh structure 212
L5K limitlow structure 212
L5K preset structure 212
L5K sbrret structure 215
L5K step structure 211
L5K stop structure 216
L5K structure 207
L5K text box structure 197
L5K transition structure 214
L5X action structure 213
L5X attachment structure 197
L5X branch structure 175
L5X condition structure 215
L5X directed link structure 217
L5X example 197, 209
L5X limithigh structure 212
L5X limitlow structure 213
L5X online edits example 209
L5X preset structure 212
L5X sbrret structure 215
L5X step structure 210
L5X stop structure 216
L5X structure 207
L5X text box structure 196
L5X transition structure 214
leg attributes 217
LOGIC 209
online edits 209
sbrret attributes 215
SFCContent 209
step attributes 211
stop attributes 216
transition attributes 214
sheet
attributes 187
guidelines 188
L5K structure 187
L5X structure 186
step
attributes 211
L5K structure 211
L5X structure 210
stop
attributes 216
L5K structure 216
L5X structure 216
structured text
instructions 231
neutral text 231
structured text routine
attributes 228
component 227
L5K example 229
L5K online edits example 229
L5K structure 227

L5X example 229
L5X online edits example 229
L5X structure 227
logic 228, 229
online edits 229
STContent 229

## T

tag 273
alarm analog attributes 115
alarm digital attributes 118
attributes 112
component 109
consumed attributes 114
coordinate system attributes 136
guidelines 142
initial values 140
L5K alarm analog example 142
L5K alarm analog structure 115
L5K alarm digital example 142
L5K alarm digital structure 118
L5K axis structure 122
L5K coordinate system structure 136
L5K example 142
L5K initial values structure 140
L5K message structure 121
L5K motion group structure 138
L5K safety example 142
L5K safety structure 139
L5K structure 110
L5X alarm analog example 142
L5X alarm analog structure 115
L5X alarm digital example 142
L5X alarm digital structure 117
L5X axis structure 122
L5X coordinate system structure 136
L5X example 142
L5X initial values structure 140
L5X message structure 120
L5X motion group structure 138
L5X safety example 142
L5X safety structure 139
L5X structure 109
motion group attributes 138
produced attributes 114
record 273
safety attributes 139
tags 30, 32
task
attributes 248
component 247
guidelines 248
L5K example 248
L5K structure 247
L5X example 248
L5X structure 247
text box
attributes 197
guidelines 218
L5K structure 197
L5X structure 196
text boxes 30, 32
text file 25
TEXTBOX 273
TEXTBOX record 275
TrackingGroup 90, 112, 173, 208, 228
transition
attributes 214
L5K structure 214
L5X structure 214
trend
attributes 255
component 255
guidelines 259
L5K example 259
L5K pen structure 259
L5K structure 255
L5X example 259
L5X pen structure 258
L5X structure 255
pen attributes 259
TXT
ALMMSG 277
comment record 275
exporting 30
format 273
importing 32
TAG 273
TXT Unicode format 273
examples 277

## U

Unicode TXT format 273

## W

watch list 263
watch tag 263
wire
  attributes 196
  guidelines 196
  L5K structure 196
  L5X structure 196

## X

XML file 28

# Rockwell Automation support

Rockwell Automation provides technical information on the web to assist you in using its products. At http://www.rockwellautomation.com/support you can find technical and application notes, sample code, and links to software service packs. You can also visit our Support Center at https://rockwellautomation.custhelp.com for software updates, support chats and forums, technical information, FAQs, and to sign up for product notification updates.

In addition, we offer multiple support programs for installation, configuration, and troubleshooting. For more information, contact your local distributor or Rockwell Automation representative, or visit http://www.rockwellautomation.com/services/online-phone.

## Installation assistance

If you experience a problem within the first 24 hours of installation, review the information that is contained in this manual. You can contact Customer Support for initial help in getting your product up and running.

| | |
|---|---|
| United States or Canada | 1.440.646.3434 |
| Outside United States or Canada | Use the Worldwide Locator available at http://www.rockwellautomation.com/locations, or contact your local Rockwell Automation representative. |

## New product satisfaction return

Rockwell Automation tests all of its products to ensure that they are fully operational when shipped from the manufacturing facility. However, if your product is not functioning and needs to be returned, follow these procedures.

| | |
|---|---|
| United States | Contact your distributor. You must provide a Customer Support case number (call the phone number above to obtain one) to your distributor to complete the return process. |
| Outside United States | Please contact your local Rockwell Automation representative for the return procedure. |

## Documentation feedback

Your comments will help us serve your documentation needs better. If you have any suggestions on how to improve this document, complete the feedback form, publication RA-DU002.

Rockwell Otomasyon Ticaret A.Ş., Kar Plaza İş Merkezi E Blok Kat:6 34752 İçerenköy, İstanbul, Tel: +90 (216) 5698400

**www.rockwellautomation.com**

**Power, Control and Information Solutions Headquarters**

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
Europe/Middle East/Africa: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
Asia Pacific: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Rockwell Automation Publication 1756-RM084Y-EN-P - December 2018

**Supersedes Publication 1756-RM084X-EN-P - February 2018**

**Copyright © 2018 Rockwell Automation Technologies, Inc. All Rights Reserved. Printed in the U.S.A.**